MASTER EN SCIENCES INFORMATIQUES

Surveillance intelligente des réseaux informatiques
conception et expérimentation d'un outil sur le réseau des Facultés

Dandoy, Stéphane

Award date:
1997

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE DAME
DE LA PAIX DE NAMUR

Institut d'informatique
Rue Grandgagnage 21, 5000 Namur

Surveillance intelligente des réseaux informatiques : conception et expérimentation d'un outil sur le réseau des Facultés

Stéphane DANDOY

Promoteurs : Messieurs les professeurs B. Le Charlier et N. Habra

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en informatique (cycle de 2 ans curriculum B).

Année académique 1996-97

Remerciements

Dès à présent je voudrais adresser mes remerciements à toutes les personnes qui ont contribué au bon déroulement de ce mémoire :

Je remercie vivement Messieurs les professeur B. Le Charlier et N. Habra qui m'ont donné la possibilité de réaliser ce mémoire et m'ont conseillé dans la rédaction de celui-ci.

Un grand merci à Aziz Mounji dont les conseils et l'attention m'ont été d'une aide précieuse.

Merci à Bruno Delcourt pour son aide lors de l'implémentation sur le réseau.

Je tiens également à remercier Valérie qui m'a beaucoup aidé dans la rédaction et la correction de ce mémoire.

Table des matières **allégée.**

Table des matières **détaillée.**

Résumé

Alors que les ordinateurs deviennent de plus en plus nombreux, de plus en plus petits et de moins en moins chers, les gens sont devenus de plus en plus intéressés par les moyens leur permettant de se connecter ensemble pour former des réseaux. Malheureusement, l'utilisation des réseaux est la porte ouverte aux problèmes de sécurité : espionnage, attaques, etc.

Un des objectifs de ce mémoire est de concevoir et d'implémenter **un outil de sécurité** permettant de détecter des schémas d'attaques. Cet outil va être développé à l'aide d'**ASAX** ; ASAX est un logiciel dont le but est de définir un outil d'analyse se basant sur un fichier séquentiel de trace et sur un langage de requêtes appelé RUSSEL (Rule-Based Sequence Evaluation Language).

Une fois l'outil développé, il sera implémenté au centre de calcul comme outil de sécurité pour le réseau des facultés.

Ce mémoire est divisé en **trois grande parties** :

•La **première partie** est consacrée à une révision des concepts importants à la bonne compréhension des schémas d'attaques présentés dans la deuxième partie.

•La **deuxième partie** est consacrée à la présentation des principes de sécurité nécessaires à la réalisation de ce mémoire : schémas d'attaques, grands principes de fonctionnement des firewalls, présentation d'ASAX, topologie du réseau des facultés.

•La **troisième partie** est consacrée à la réalisation proprement dite : programmation et implémentation à l 'aide d'ASAX et RUSSEL.

Abstract

As computer have become smaller, cheaper, and more numerous, people have become more and more interested in connecting them together to form networks. Unfortunately, networks can be the source of many troubles : spying, hacking, etc.

One of this thesis objectives' is to conceive and implement a security tool enabeling attack schemes' detection. This tool is developed with ASAX (Advanced Security Audit Trail Analysis on Unix); the goal of the ASAX software is to define an analysis tool based on a sequential audit-trail file and on a rule based language named RUSSEL (Rule_Based Sequence Evaluation Language).

Once developed, this tool will be implemented as Faculties' network security tool.

This thesis is divided into **three parts** :

•**First**, we will remind you what you have to know to understand the second part of this thesis : mainly TCP/IP protocol suite.

•**Second**, we will present important security principles and concepts used in this thesis : attack schemes, firewalls, ASAX.

•**Finally**, we will show how was developed this tool with ASAX and RUSSEL : programs and implementations.

Introduction

Les réseaux,...

Alors que les ordinateurs deviennent de plus en plus nombreux, de plus en plus petits et de moins en moins chers, les gens sont devenus de plus en plus intéressés par les moyens leur permettant de connecter leur ordinateurs ensemble pour former des réseaux ou des systèmes distribués.

Après l'informatique *centralisée,* nous avons connu une période de développement intense de l'informatique *personnelle* : station de travail et surtout micro-ordinateurs. Cette nouvelle optique a bouleversé les façons de faire et les organisations informatiques traditionnelles. Mais cette informatique *personnelle* s'est révélée souvent trop individualiste pour s'intégrer harmonieusement à des organisations importantes.

Si les inconvénients de la centralisation de l'informatique sont connus, les avantages de la centralisation de l'information sont évidents : possibilité de garantir en pratique l'intégrité de l'information quand elle est dupliquée sur de nombreux systèmes incompatibles ou qui communiquent mal entre eux.

Les réseaux locaux se sont alors développés pour résoudre ces problèmes. L'informatique est aujourd'hui très souvent distribuée, ce qui permet de bénéficier simultanément des avantages de la *centralisation* de l'information et de la *décentralisation* des moyens, et des atouts que procure une circulation rapide de l'information entre tous ses utilisateurs.

...la sécurité,...

> It's easy to run a secure computer system. You merely have to disconnect all dial-up connections and permit only direct-wired terminals, put the machine and it's terminals in a shielded room, and post a guard at the door.
>
> R.H. Morris[1]

Comme on vient de le voir, et que ce soit pour le meilleur ou pour le pire, la plupart des systèmes informatiques actuels ne sont pas configurés de cette façon. La sécurité est, en général, le résultat d'un compromis. En effet, la plupart des gens ne

[1] R.H. Morris est l'auteur du Ver d'Internet (Internet Worm) qui sema la panique dans le monde des réseaux le 2 novembre 1988.

sont pas désireux de renoncer aux avantages que représentent les accès à distance (via réseau). Ils vont dès lors inévitablement subir une diminution de leur niveau de sécurité.

L'utilisation des réseaux est dangereuse pour au moins deux raisons :

- Les réseaux augmentent le nombre de points à partir desquels une **attaque** peut être lancée.
- Les réseaux augmentent le périmètre de votre système informatique à des machines qui sont sous le contrôle de personnes auxquelles il ne faut pas, *a priori*, faire confiance. Les informations reçues ne sont donc plus forcément authentiques et il convient donc de prendre des précautions.

Les ordinateurs placés en réseau possèdent une autre particularité qu'il convient de noter : la plupart des organisations possèdent un certain nombre d'ordinateurs qui sont connectés entre eux et au monde extérieur. Ceci constitue à la fois un avantage et un inconvénient : un inconvénient, car il faut que les machines puissent avoir confiance les unes en les autres, et un avantage car il est possible de configurer le réseau local de façon à ce qu'une seule machine de l'organisation soit en contact avec le monde extérieur. Il est alors possible de focaliser tout ce qui concerne la sécurité en un seul point. Ce dispositif est plus connu sous le nom de **Firewall**.

...,et les Facultés.

L'institut d'informatique possède un réseau d'ordinateurs. Ce réseau fait lui-même partie du réseau des Facultés et comme de bien entendu, celui-ci est relié à Internet. Nous nous trouvons donc dans la situation décrite ci-dessus : nous disposons d'un réseau relié à Internet et dont nous bénéficions des avantages mais également des inconvénients.

Parmi toutes les solutions qui existent en matière de protection des réseau, celle choisie dans ce mémoire est celle de l'**audit**[2]. Au sein de l'institut d'informatique a été développé un programme portant le nom d'ASAX (Advanced Security Audit Trail Analysis on Unix). Celui-ci a pour objectif de définir un outil d'analyse se basant sur un fichier séquentiel de trace et sur un langage de requêtes appelé RUSSEL (Rule-Based Sequence Evaluation Language).

Le but de ce mémoire est d'exploiter ASAX pour en faire un outil de sécurité nous permettant de faire bénéficier le réseau informatique des faculté un peu plus des avantages que des inconvénients.

[2] Les différentes solutions qui existent en matière de protection des réseaux seront présentées dans la partie consacrée à la sécurité.

En effet, le réseau des Facultés tel qu'il existe à l'heure actuelle ne bénéficie d'aucune protection face à d'éventuelles attaques externes. Un des objectifs de ce mémoire va être de réaliser un pseudo-firewall permettant de détecter des schémas d'attaques. Il ne s'agit donc pas ici d'automatiser les mesures à prendre à l'encontre des pirates mais simplement de détecter les attaques éventuelles afin de décider si oui ou non il convient de réagir.

La raison pour laquelle c'est cette stratégie qui a été adoptée, est que nous nous trouvons dans un environnement universitaire et qu'il convient avant tout de préserver une ouverture sur le monde et donc sur l'Internet.

*La **première partie** de ce mémoire est consacrée a un rappel des bases essentielles en matières de réseaux : suite TCP/IP, Ethernet[3], etc. Il s'agit en fait de prérequis permettant une bonne compréhension de ce qui suit. Il est évident que, pour les connaisseurs, cette partie peut être passée.*

*La **deuxième partie** quant à elle présente les différents aspects de la sécurité qui ont une importance pour ce travail : étude de ce qui peut se faire en matière d'attaque sur les réseaux, présentation de ASAX et des firewalls, description de l'installation réseau des facultés.*

*La **troisième partie** sera consacrée à la réalisation et à l'implémentation proprement dite.*

[3] La suite TCP/IP et Ethernet sont les protocoles utilisés aux Facultés. Ces deux protocoles sont, de plus, ceux qui sont les plus employés dans le monde. Le fait que ce soit justement sur ces deux types de protocoles que porte ce mémoire va permettre de donner à celui-ci un caractère tout à fait global et applicable à d'autres sites.

Première Partie : Prérequis.

1 Introduction et concepts de base[1].

1.1 Evolution vers les réseaux.

Après l'informatique *centralisée,* nous avons connu une période de développement intense de l'informatique *personnelle* : station de travail et surtout micro-ordinateurs. Cette nouvelle optique a bouleversé les façons de faire et les organisations informatiques traditionnelles. Mais cette informatique *personnelle* s'est révélée souvent trop individualiste pour s'intégrer harmonieusement à des organisations importantes.

Si les inconvénients de la centralisation de l'informatique sont connus, les avantages de la centralisation de l'information sont évidents : possibilité de garantir en pratique l'intégrité de l'information quand elle est dupliquée sur de nombreux systèmes incompatibles ou qui communiquent mal entre eux.

Les réseaux locaux se sont alors développés pour résoudre ces problèmes. L'informatique est aujourd'hui très souvent distribuée, ce qui permet de bénéficier simultanément des avantages de la *centralisation* de l'information et de la *décentralisation* des moyens, et des atouts que procure une circulation rapide de l'information entre tous ses utilisateurs.

L'hétérogénéité d'un parc informatique, malgré ses inconvénients, est le meilleur moyen de garantir une évolution sans à-coup et un fonctionnement correct de la concurrence entre fournisseurs.

1.2 Avantages des réseaux.

1.2.1 Communication facile et rapide de l'information.

Particulièrement importante dans le domaine de la recherche qui a vu naître les grands réseaux, la communication rapide et à grande échelle de l'information est indispensable à toute organisation dont la taille dépasse le groupe d'individus. Une organisation répartie sur plusieurs sites distants ne peut se passer d'un réseau d'interconnexion, quelle que soit son activité.

[1] La plupart des notions introduites dans cette introduction seront revues plus en détails par la suite.

1.2.2 Partage des ressources : matérielles, logicielles, données,...

La mise en commun de ressources matérielles (imprimantes, espaces disque, périphériques coûteux, calculateur puissant utilisé de manière épisodique) est une puissante motivation à la mise en réseau.

La mise en commun de ressources logicielles procède de la même logique, et une licence logicielle, comme une imprimante, peut être partagée.

La mise en commun de données est un point essentiel au bon fonctionnement d'une organisation. La centralisation et le partage de l'information permettent d'éviter les incohérences de duplication. La mise à jour non simultanée des différentes copies conduit en effet à des erreurs d'exactitude de l'information pour certains utilisateurs. Or, composante essentielle de l'intégrité, c'est un élément clef de tout système d'information.

Outre l'économie des moyens techniques et l'amélioration de l'intégrité de l'information, cette mise en commun de ressources procure d'importantes économies de moyens humains : mise à jour d'un jeu de données ou d'un logiciel, effectuée une fois, est immédiatement prise en compte par tous les utilisateurs de toutes les machines du réseau.

1.2.3 Accès à des outils informatiques adaptés au problème du moment.

Il s'agit simplement d'optimiser l'investissement en moyens informatiques. Par exemple, un utilisateur effectuant d'importants calculs statistiques sur de grands jeux de données peut avoir besoin d'un tableur pour visualiser ses résultats. L'utilisateur pourrait employer deux machines pour ces deux tâches différentes. Si l'accès transparent à l'information est assuré, la mise en réseau de machines de puissances très différentes permet d'utiliser chacune de façon optimale, sans inconvénient pour l'utilisateur.

1.3 Principes des applications réseaux.

1.3.1 Les différents types d'applications.

Les applications réseau représentent les fonctionnalités offertes par un réseau de machines. Elles peuvent être classées en cinq catégories présentées ci-dessous. Ces applications, dont les noms sont indiqués entre parenthèses, seront, pour la plupart, détaillées dans ce chapitre, dans la partie consacrée aux protocoles.

1.3.1.1 Transfert de fichiers entre machines distantes (**ftp, tftp, rcp**).

Un transfert de fichier réalisé entièrement par voix électronique entre machines distantes évite tout problème d'incompatibilité et de manipulation de média, et améliore les délais d'échanges de fichiers.

1.3.1.2 Connexion sur un ordinateur distant (**telnet, rlogin, rsh**).

L'utilisateur d'une machine A peut transformer son poste de travail en terminal d'une machine B. Travaillant en environnement multi-fenêtres, un utilisateur pourra ouvrir des sessions simultanées sur des machines différentes sans quitter son poste de travail.

1.3.1.3 Courrier électronique (**mail, talk**).

De type Courrier électronique (**mail**) ou conversation interactive (**talk**) , le courrier permet de gérer facilement des listes de diffusion. Si de plus il permet d'attacher à un message des documents non textuels (binaires, ...), c'est un outil de communication inégalable.

1.3.1.4 Accès transparent à des fichiers distants (**nfs, rfs**).

Fonctionnalité très supérieure au simple transfert de fichiers cité plus haut, un sous ensemble de l'arborescence de fichiers d'une machine est physiquement localisé sur le disque d'une autre machine. Du point de vue de l'utilisateur (et des logiciels utilisant ces fichiers), rien ne distingue ces fichiers distants des fichiers locaux. Cette fonctionnalité permet la mise en commun d'espace disque, mais surtout de logiciels ou de données. Cette même application permet l'accès transparent à des imprimantes distantes.

1.3.1.5 Traitement réparti (**ncs**).

Cette fonctionnalité permet de partager la puissance de traitement disponible sur les machines du réseau. La partage de la puissance de calcul sur le réseau est au stade final du laboratoire chez tous les grands constructeurs.

1.3.2 Le concept de Client-Serveur.

Les applications réseau reposent sur le concept de client-serveur. Des processus clients utilisent sur le réseau des services assurés par des processus serveur. Les requêtes du client sont normalisées, de même que les réponses du serveur. En d'autres termes, client et serveur sont les deux moitiés d'une application réseau, et son conçus pour travailler ensemble via le réseau. Le client prend l'initiative de la communication. Le serveur répond aux requêtes du client.

Il serait erroné de penser qu'un serveur (ou un client) est une machine : c'est un processus. Sur une machine (p.e. : Unix)s'exécutent simultanément **des** clients **et des** serveurs. Il est également faux de croire que le serveur fournit les données et que le client les reçoit. Prenons l'exemple d'un transfert de fichier par **ftp** entre la machine α

et la machine β : un utilisateur sur α envoie un fichier sur la machine β. Dans ce cas le client ftp de α se connecte au serveur ftp de β et lui envoie le fichier. Le serveur reçoit le fichier. Pendant ce temps, d'autres applications réseau (ou la même) entre α et β pourraient mettre en action d'autres serveurs et d'autres clients sur les deux machines.

Il est également faux d'associer serveur à grosse machine et client à petite machine. En effet, lorsqu'un utilisateur se connecte sur un ordinateur à partir d'un terminal X, le serveur X, qui exécute les opérations graphiques, est dans le terminal X. Les clients X, applications demandant un affichage graphique, s 'exécutent sur l'ordinateur.

1.4 Classification des réseaux[2].

Il n'est pas inutile de définir quelques termes fréquemment rencontrés dans le vocabulaire des réseaux, et qui sont parfois sources d'ambiguïté.

On peut tout d'abord **classer** les réseaux **suivant leur taille** :

1°- Le Réseau Local (***LAN*** *: Local Area Network*) désigne un réseau de faible extension interconnectant sur un même medium quelques dizaines de machines. Un tel réseau est limité, dans le cas général, à un bâtiment ou un faible nombre de bâtiments. C'est le cas du réseau Ethernet.

2°- Le Réseau Métropolitain (***MAN*** : *Metropolitan Area Network*) est un réseau plus étendu, constitué de plusieurs réseaux locaux interconnectés par des liaisons point à point dont les débits sont très souvent identiques à ceux des réseaux locaux.

3°- Le Réseau à Grande Echelle (***WAN*** : *Wide Area Network* ou Long Haul Network) désigne un réseau constitué de plusieurs réseaux des types précédents, interconnectés par des réseaux de télécommunications publics. C'est le cas , par exemple du réseau Internet, d'extension planétaire, et sur lequel sont connectés plus d'un million de machines, ou de réseaux ***EARN*** ou ***Bitnet***.

Les frontières entre ces différentes échelles ne sont pas toujours très claires, mais les définitions précédentes s'appliquent à la majeure partie des cas.

Les applications réseau, en tout état de cause ne seront pas les mêmes suivant l'échelle : si le transfert de fichier, l'émulation de terminal et le courrier électronique s'utilisent sur des réseaux à large échelle, le partage de fichier et la calcul réparti ne se font habituellement que sur des réseaux locaux ou métropolitain.

On peut également **classer** les réseaux en **réseaux logiques** ou réseaux **physique** : pour l'utilisateur, l'appartenance au réseau Internet peut être pour lui une réalité plus immédiate que la limite de son propre réseau local.

[2] Voire figure 1 page suivante.

La figure 1 (ci-dessous) reprend les différentes classes de réseaux. On y développe également la branche des réseaux de transmission à grande distance (*WAN*). Dans le cadre de ce travail de fin d'étude, c'est précisément cet aspect de la télécommunication qui nous intéresse. En effet, nous n'allons pas nous concentrer sur les données circulant à l'intérieur du réseau[3] (p.e. : le réseau des facultés) mais bien sur les données échangées avec l'extérieur.

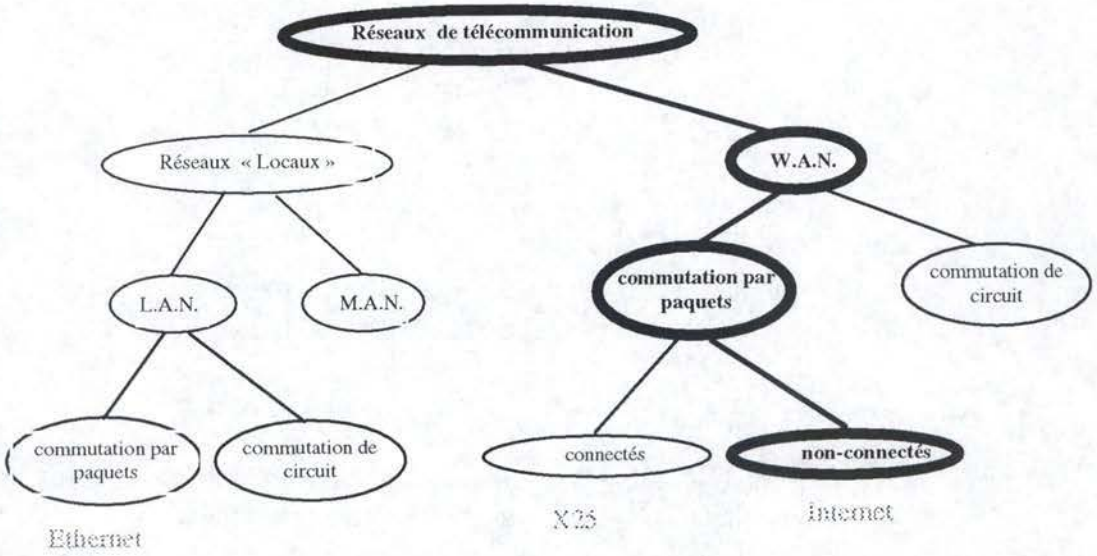


Figure 1 : hiérarchie de décomposition des réseaux de télécommunication. En gras, les réseaux qui nous intéressent plus particulièrement dans ce travail. C'est le cas, par exemple, d'Internet.

Bien que tous les réseaux de transmission à longue distance permettent des connexions entre deux ordinateurs ou entre un ordinateur et un terminal, on peut les diviser en deux types de base : les **réseaux à commutation de circuit** et les **réseaux à commutation par paquets** (un troisième type existe, la **commutation par messages** mais nous n'en feront pas état ici car fort spécialisée et hors de propos).

1° Les réseaux à commutation de circuits.

Les réseaux à commutation de circuits opèrent en réalisant un circuit physique entre deux points à travers le réseau. Ce circuit est maintenu et utilisé exclusivement et en permanence par les deux interlocuteurs pendant toute la durée de la communication. La liaison physique est donc totalement réservée à l'échange même pendant les périodes de silence. Le réseau téléphonique (RTC : Réseau Téléphonique Commuté), par exemple, utilise la technique de commutation de circuit - un appel téléphonique établit un circuit du téléphone source, à travers les noeuds du réseau jusqu'au téléphone destinataire. Quand un circuit est établi, aucun autre trafic ne peut circuler sur les fils formant ce circuit.

[3] Cfr. Travail de fin d'étude de Borhème Regaya : « Développement d'un outil de surveillance et de détection d'attaques sur réseaux Ethernet ».

- L'**avantage** de cette technique réside dans **l'assurance de capacité** : une fois le circuit établi, aucune autre activité du réseau ne peut faire décroître la capacité du circuit.
- Un **désavantage** est le **coût** : les coût du circuit sont fixes et indépendant du trafic. Le tarif est fixe et indépendant de la rapidité avec laquelle les deux parties communiquent.

2° Les réseaux à commutation par paquets.

Cas général.

Les réseaux à commutation par paquets, généralement employés pour la communication entre ordinateurs, reposent sur une approche tout à fait différente. Dans les réseaux à commutation par paquets, les flux de données sur le réseau sont divisés en petits segments appelés paquets qui sont multiplexés sur des connexions à haute capacité. Un paquet, qui contient seulement quelques centaines de bytes de données, contient des données d'identification permettant aux ordinateurs sur le réseau de savoir si le paquet leur est destiné ou comment le faire parvenir au destinataire. Par exemple, un fichier à transmettre entre deux machines peut être découpé en plusieurs paquets qui sont envoyés à travers le réseau un à un. Le réseau délivre les paquets à la destination spécifiée où le software réseau ré-assemble le tout en un seul fichier à nouveau.

- L'**avantage** principal de cette technique est que de multiples communications entre machines peuvent se dérouler en concurrence, les connexions entre machines pouvant être partagées par toutes les paires de machines en train de communiquer

- Un **désavantage** est, bien sûr, que quand le trafic augmente sur le réseau, une paire donnée d'ordinateurs en train de communiquer peut utiliser de moins en moins de capacité de la ligne.

Malgré ce désavantage, les réseaux à commutation par paquets sont devenus extrêmement populaires. Les motivations pour adopter cette technique sont le coût et la performance. Puisque de nombreuses machines peuvent se partager le réseau, le nombre d'interconnexions requis diminue et donc le coût peut être maintenu assez bas. Les ingénieurs ayant su développer des hardware réseau à grande vitesse, la capacité n'est pas un problème. **Lorsque l'on utilisera le terme de réseau par la suite, on sous-entendra généralement un réseau à commutation par paquet** .

Cas particuliers.

Comme le montre la figure 1, les réseaux communicant suivant la technique de commutation par paquets peuvent le faire suivant deux modes : communication par paquets en mode **connecté** (les réseaux X25) ou en mode **non-connecté** (Internet).

1° Dans le **mode connecté** (*connection-oriented*), on va essayer d'ajouter les qualités des RTC à la technique de commutation par paquets, à savoir :

- protection contre les **erreurs de transmission** ;

- pas de **perte** de paquet ;
- arrivée des paquets en **séquence** à la destination;
- élimination du **dédoublement** des paquets ;

Pour assurer ces quatre propriétés, on va imiter le mode de communication par commutation de circuits en créant un *circuit virtuel*. La création d'un circuit virtuel se fait par l'envoi de paquets d'appel à travers tous les noeuds du réseau jusqu'à la destination et, inversement, par l'envoi de paquet de confirmation de la destination vers la source. Une fois établi, ce circuit virtuel va porter un numéro.

Après cela, on peut envoyer son fichier. Celui-ci sera découpé en paquet, chaque paquet portant un numéro d'ordre dans la **séquence** des paquets constituant le fichier. Chaque paquet va également porter le numéro de circuit virtuel.

Les lignes sont assurées **sans perte** et **sans erreur** grâce au protocole de niveau 2 HDLC (les protocoles sont détaillés à la fin de ce chapitre).

Enfin, en ce qui concerne l'élimination du **dédoublement**, ce sont les noeuds qui s'en chargent.

- **Avantages** : on peut en trouver des réseaux X25 presque partout et il existe de nombreux équipements compatibles.

- **Inconvénients** :
  - lenteur, due au long temps de calcul s'effectuant à chaque noeud pour le routage.
  - coût d'emploi élevé ;
  - dépassé techniquement ;

2° Dans le mode **non-connecté** (*connection-less*), aucune des quatre propriétés précitées n'est assurée. Chaque paquet contient l'adresse de l'expéditeur et du destinataire.

L'emploi des lignes digitales, et de fibres optiques diminue le risque d'erreur de transmission. Utiliser des nœuds intelligents pour le routage des paquets devient donc de plus en plus inutile. **La tendance actuelle s'oriente donc plutôt vers le mode non-connecté plutôt que vers le mode connecté. Ce sera également le cas dans ce travail.**

2 Protocoles d'Internet

Dans cette partie vont être présentés : tout d'abord la suite de protocole TCP/IP dans son ensemble et en toute généralité, ensuite, ce seront les différents protocoles de cette suite ainsi que les protocoles satellites qui seront détaillés.

2.1 TCP/IP: Définition et architecture

2.1.1 Définition

TCP/IP *(Transmission Control Protocol/Internet protocol)* est le résultat d'études de recherches financées par la DARPA *(Defense Advanced Research Projects Agency).* Cette technologie comporte un ensemble de standards pour réseaux. Ces standards spécifient les détails de communication entre ordinateurs ainsi qu'un ensemble de conventions qui régissent l'interconnexion des réseaux et le routage du trafic.

Cette technologie est appelée communément TCP/IP car le cœur de ce modèle est formé par le protocole IP *(Internet Protocol)* et par le protocole TCP *(Transmission control protocol).* Le protocole IP fournit un service de communication non fiable (*unreliable*) entre ordinateurs de l'Internet, il correspond assez bien à la couche 3 du modèle OSI. Le protocole TCP rend fiable ce service de communication, il est proche du protocole de la classe 4 de la couche transport dans l'architecture OSI.

Ces protocoles fournissent des recettes pour faire passer des messages dans le réseau, ils spécifient les détails du format des messages et décrivent des moyens pour corriger les erreurs éventuelles.

TCP/IP forme une technologie de base pour le réseau Internet (inter-connexion de réseaux) qui connecte la majorité des institutions de recherches aux Etats-Unis d'Amérique. La NASA *(National Aeronautics and Space Administration),* le NSF *(National Science Foundation)* et le département de l'énergie font partie de ces institutions. Ils utilisent TCP/IP pour connecter leurs sites de recherches avec ceux de la DARPA. La structure résultante est appelée l'Internet DARPA ou l'Internet TCP/IP ou plus simplement l'Internet. Cette connexion permet aux chercheurs de ces institutions de partager les informations avec leurs collègues à travers tout le pays comme s'ils étaient tous dans le même bureau.

Le modèle TCP/IP a été conçu pour connecter différents types de support de communication utilisant la **technique de la commutation par paquets**. Ces supports sont reliés entre eux par des passerelles (*gateways*). Ensemble, ils forment un réseau appelé l'Internet. Le modèle TCP/IP fournit un service fiable de communication entre processus s'exécutant sur des ordinateurs hôtes attachés à l'Internet.

Les réseaux de communication les plus développés dans les systèmes Unix sont basés sur TCP/IP. Les liaisons physiques locales sont réalisées à partir de réseaux Ethernet pour la plupart (ce qui est le cas des facultés).

2.1.2 Bref Historique

Au début des années 80, le département de la défense américaine (DoD) a mis au point les protocoles TCP/IP, précurseur des modèles basés sur les couches de communication. Ces protocoles de communication sont le fruit d'un projet, lancé au début des années 70, destiné à standardiser les moyens de communication entre les ordinateurs de l'administration de la défense américaine. Le groupe de recherche qui a mené ce projet était le premier à introduire le concept de couches de communication. Le résultat de ce travail était l'élaboration des protocoles du réseau ARPANET. Celui-ci était suivi par le développement de TCP/IP dont les premières implémentations ont vu le jour en 1980. Cette série de protocoles a bénéficié des innovations du moment, tels les réseaux à commutation de paquets, les réseaux à collision ainsi que diverses technologies devenues opérationnelles. L'université de Berkeley, les a intégrée dans son Unix BSD favorisant ainsi la généralisation de son utilisation. Il n'en reste pas moins que TCP/IP était le précurseur des modèles basés sur les couches de communication, modèle que le standard OSI a plus tard consacré. Mais TCP/IP reste le plus ancien et aussi le plus simple.

Au début des années 90, vu le nombre croissant de réseaux, un nouveau problème se présente pour TCP/IP : son espace d'adressage est limité. Cependant, le modèle OSI offre une plus grande souplesse. Malgré ceci, le modèle TCP/IP est toujours relativement utilisé, tandis que le modèle OSI se répand de plus en plus.

Dans ce qui suit, nous parlerons des différents protocoles de façon plus ou moins détaillée; puis nous présenterons l'architecture conceptuelle du modèle et de son organisation en couches.

2.1.3 Principes et fonctionnement de TCP/IP

2.1.3.1 Introduction

Les systèmes complexes de communication de données utilisent plus d'un protocole pour effectuer toutes les tâches de transmission, ils ont besoin d'un ensemble de protocoles coopératifs, appelés familles de protocoles.

Les problèmes qui peuvent surgir, quand les machines communiquent entre elles sur un réseau de données, sont de plusieurs types :

- **Panne du matériel:** un hôte ou une passerelle peut tomber en panne: bris de matériel ou dysfonctionnement du système d'exploitation. Une liaison du réseau de transmission peut tomber en panne ou peut être déconnectée accidentellement. Le système doit détecter une telle panne et si possible la réparer.

- **Encombrement du réseau:** même si le matériel et les logiciels fonctionnent correctement, les réseaux ont une capacité finie qui ne peut être dépassée. Le système doit gérer l'encombrement pour que le trafic reste fluide.

- **Perte ou retard des paquets:** parfois, les paquets sont perdus ou arrivent en retard. Le logiciel du protocole doit connaître les raisons de la perte ou s'adapter aux retards.

- **Donnée corrompue:** les interférences magnétiques ou électriques peuvent causer des erreurs de transmission qui vont corrompre le contenu des données transmis. Le système doit détecter et corriger de telles erreurs.

- **Erreurs de duplication ou de séquence:** les réseaux qui offrent des routes multiples peuvent délivrer les données en désordre. Le système doit ré-ordonner les paquets à l'arrivée et supprimer les doubles.

Avec tous ces problèmes et vu la diversité des tâches, il devient difficile d'écrire un seul protocole qui peut gérer le tout.

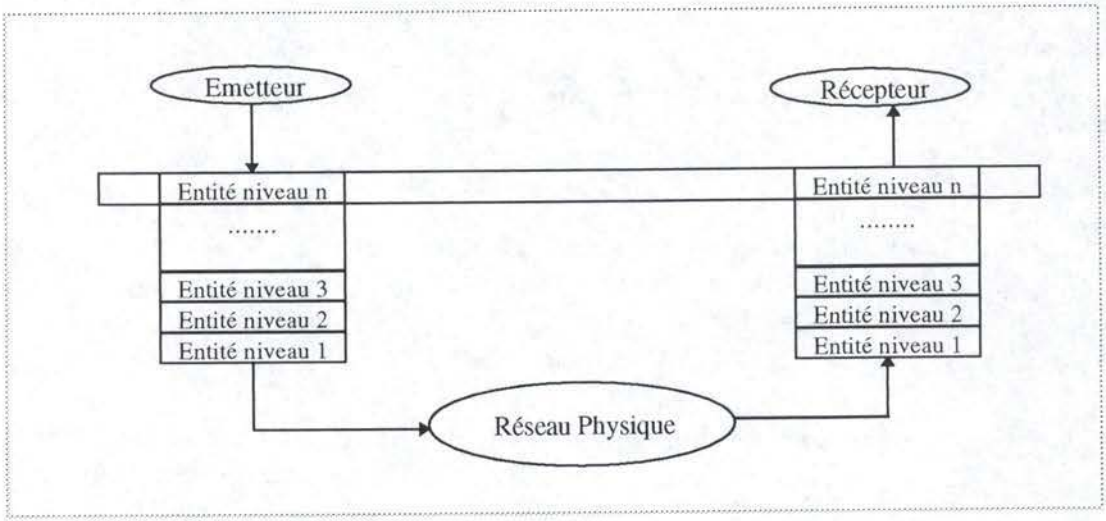
Par analogie aux langages de programmation, la division en sous problèmes donne au concepteur la possibilité de se concentrer sur une chose à la fois, et à celui qui fait l'implémentation de construire et tester chaque partie du logiciel indépendamment.

2.1.3.2 Architecture d'une famille de protocoles: les couches conceptuelles

Sur chaque machine, les modules du logiciel du protocole (le système) sont empilés verticalement en forme de couches. Chaque couche est responsable de la manipulation d'une partie du problème. Entre les couches, il existe a une relation UTILISE qui a pour objectif de cacher les détails à l'utilisateur.

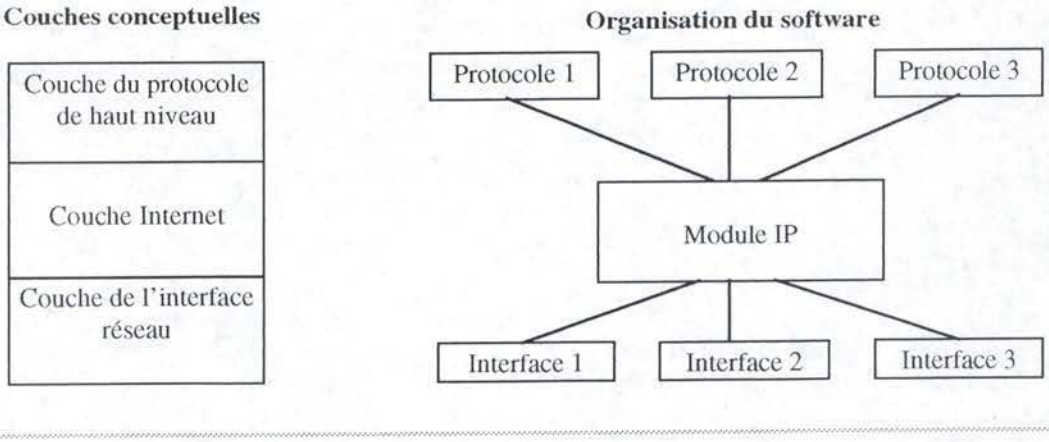
Généralement, transmettre un message d'un programme d'application sur une machine à un programme d'application sur une autre machine signifie que le message passe de haut en bas à travers les entités successives du logiciel du protocole de la machine émettrice, puis traverse le réseau, et enfin remonte de bas en haut à travers les entités successives du logiciel du protocole de la machine réceptrice.

Ci-dessous une figure de l'organisation du logiciel en couches de familles de protocoles:



Les logiciels qui implémentent une famille de protocoles sont très complexes. Chaque couche prend des décisions dans la correction du message et choisit une action appropriée basée sur le type de message ou sur l'adresse de destination. Par exemple, une couche sur la machine réceptrice doit décider de garder le message ou de le réexpédier vers une autre machine. Une autre couche doit décider quelle application devrait recevoir le message, etc.

La différence entre l'organisation conceptuelle du logiciel du protocole et les détails d'implémentation est que dans le diagramme conceptuel, comme le montre la figure ci dessous, on considère une couche appelée Internet entre les entités de protocole de haut niveau et l'entité de l'interface réseau, par contre le diagramme réaliste considère que le logiciel IP peut communiquer avec plusieurs entités (modules) de protocoles de haut niveau et avec plusieurs interfaces réseau.

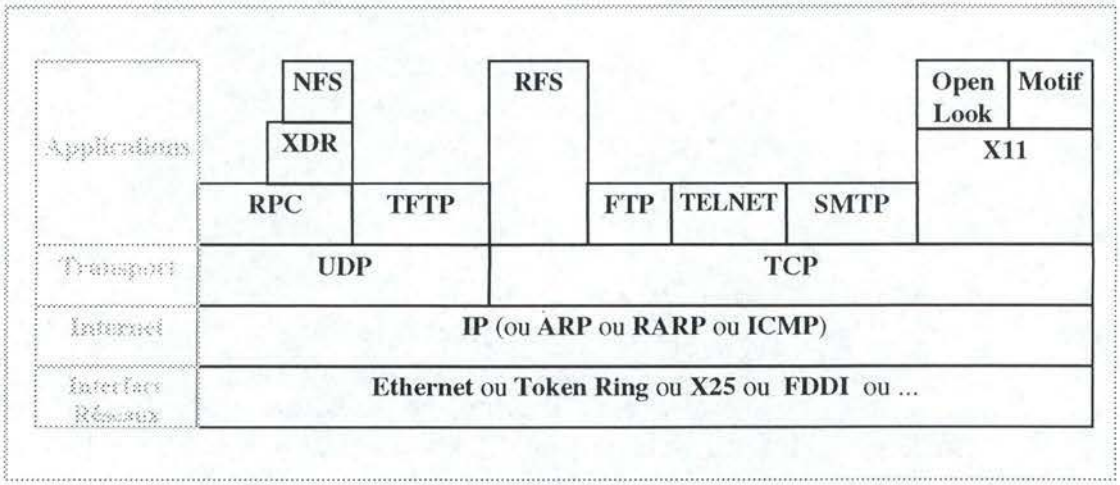


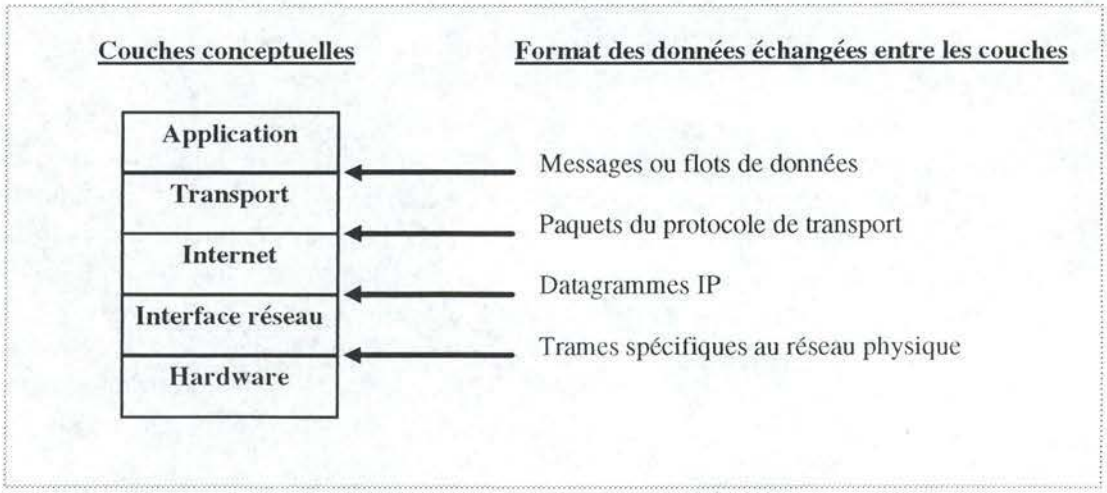
En fait, le modèle conceptuel a pour but de décrire les services offerts par chaque couche. Il n'a pour but de décrire leur fonctionnement.

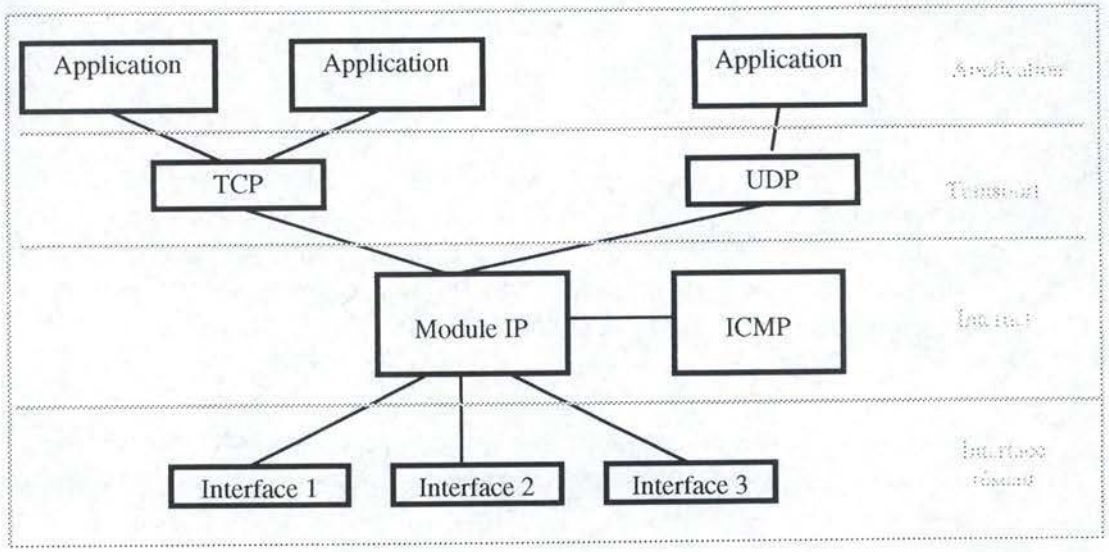
2.1.3.3 Architecture en couches du modèle TCP/IP

Une fois qu'on a pris la décision de partitionner le problème de la communication en plusieurs sous-problèmes et que l'on a organisé le logiciel de protocole en modules, chaque module va s'occuper d'un problème particulier et va occuper un emplacement précis appelé couche. La structure en couches va être également utilisée ultérieurement par le modèle OSI.

Le logiciel de TCP/IP est organisé en quatre couches conceptuelles qui sont construites sur une cinquième couche propre au matériel *(hardware)*. Les deux figures ci-dessous montre les couches conceptuelles ainsi que les formes de données qui passent entre ces couches.







2.1.3.3.1 Couche interface réseau:

La couche la plus basse du logiciel du protocole Intenet est la couche d'interface réseau. Cette couche reçoit des datagrammes IP et les transmet sur un réseau spécifique (sous-réseau). Une interface réseau peut être composée d'un dispositif conducteur ou d'un sous-système complexe qui utilise son propre protocole de liaison de données.

2.1.3.3.2 Couche Internet:

La principale mission de cette couche est d'obtenir l'interconnexion des différents sous-réseaux. Pour y arriver, un mécanisme unique de transport de données a été choisi. Ce mécanisme est le protocole IP. Le protocole IP regroupe toutes les fonctions nécessaires pour transmettre une suite d'octets, appelée datagramme, entre un ordinateur hôte source et un ordinateur hôte destination connectés éventuellement à des sous-réseaux distincts. Il ne dispose, toutefois, pas de mécanismes qui assure la fiabilité de la transmission des données de bout en bout, ou qui contrôlent le flux des datagrammes, ou encore qui règlent le séquencement des datagrammes. Le service offert par le protocole IP est donc non fiable (*unreliable*).

Le protocole IP implémente trois fonctions de base : l'**adressage**, le **routage** et la **fragmentation**. L'adresse Intenet est une adresse de longueur fixe. Elle permet d'identifier les ordinateurs hôtes et les passerelles de l'Intenet. Cette adresse Internet est utilisée pour acheminer les datagrammes à leur destination. Quand un datagramme doit traverser un sousréseau dont la taille maximale du champ des données des paquets est inférieure à la taille du datagramme, le protocole IP fragmente le datagramme en fragments de taille plus petites. A l'arrivée, le datagramme initial est réassemblé avant d'être fourni à l'utilisateur. Le protocole IP traite chaque nouveau datagramme de manière indépendante. Ils pourront donc suivre des itinéraires différents avant d'arriver à leur destination.

La couche IP s'occupe donc des communications entre machines. L'entité IP de la machine source accepte une demande d'envoi d'un paquet d'une couche transport avec l'identification de la machine à laquelle le paquet **devrait** être transmis. Elle encapsule le paquet dans un datagramme IP, remplit l'en-tête du datagramme, utilise l'algorithme de routage pour déterminer si elle doit le livrer directement ou l'envoyer à une passerelle, elle passe le datagramme à l'interface du réseau appropriée pour la transmission. L'entité IP de la machine de destination s'occupe des datagrammes reçus, elle vérifie leur validité, supprime l'en-tête, et utilise l'algorithme de routage pour décider si le datagramme devrait être traité localement ou réexpédié. Pour les datagrammes destinées à une machine située sur le même sous-réseau que la passerelle, le logiciel de la couche Internet choisit parmi plusieurs protocoles de transport celui qui va recevoir le paquet.

L'encapsulation signifie que TCP, par exemple, crée un en-tête pour la donnée que l'utilisateur désire transmettre, ce datagramme est passé à la couche IP. La couche IP crée également un en-tête de ce qu'elle reçoit de TCP. Finalement, la couche interface du réseau met le datagramme dans une trame avant de le transmettre d'une machine à une autre. Le format de la trame dépend de la technologie du réseau utilisé.

A l'entrée, un paquet arrive à la couche la plus basse du logiciel du réseau et commence son ascension à travers les couches successives. Chaque couche enlève un en-tête avant de passer le message à la couche suivante de façon à ce que la couche la plus haute passe seulement les données au processus récepteur, sans aucun en-tête. Ainsi, l'en-tête le plus externe correspond à la plus basse couche du protocole, alors que l'en-tête le plus interne correspond à la plus haute couche du protocole.

Enfin, la couche Internet envoie des messages ICMP *(Internet Control Message Protocol) et* s'occupe de tous les messages ICMP reçus. Ce protocole (ICMP) permet d'envoyer des messages de contrôle et des messages d'erreur entre sites de l'Internet. Il est également chargé de renseigner l'ordinateur hôte source des problèmes qui surviennent aux datagrammes lors de leur transmission. Il peut également fournir à l'ordinateur hôte source des conseils ou des renseignements concernant le fonctionnement de l'Internet. Le protocole ICMP ne garantit toutefois pas que, si le datagramme n'a pas été livré à l'ordinateur hôte de destination, l'ordinateur hôte source en sera informé. Seuls les problèmes détectés par le protocole IP et pour lesquels un message ICMP est prévu, seront communiqués à l'entité IP émettrice.

2.1.3.3.3 Couche transport:

La première fonction de la couche transport est de fournir une communication entre deux programmes d'application situés sur des machines hôtes distinctes, au travers du réseau. Une telle communication est souvent appelée "de bout en bout". La couche de transport peut régulariser le flux d'information, elle peut également fournir un service de transport fiable, assurant une arrivée de données sans erreurs et dans l'ordre. Pour ce faire, elle reçoit des accusés de réception de la machine de destination et retransmet les paquets perdus. Le logiciel de transport divise le flot de données qui va être transmis en des petites pièces appelés paquets (suivant la terminologie OSI). Chaque paquet est transmis avec son adresse de destination au niveau suivant.

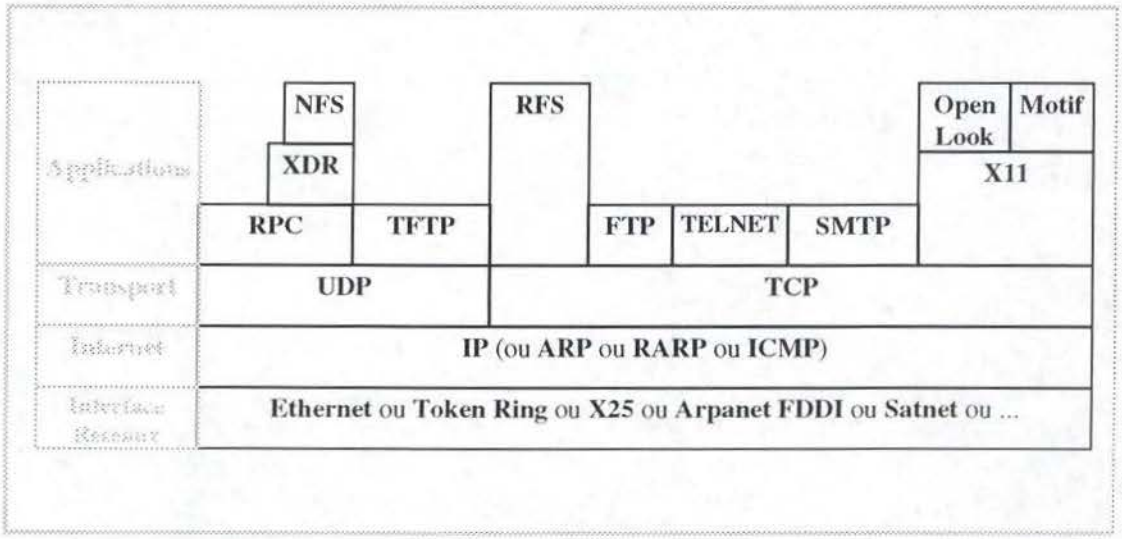
Dans la figure représentant le modèle des couches conceptuelles on utilise un seul bloc pour représenter la couche application. Cependant, un ordinateur puissant peut avoir plusieurs programmes d'application qui peuvent communiquer avec d'autres sur machines différentes, via le réseau, en même temps. La couche transport doit accepter les données de plusieurs programmes utilisateurs et les transmet à l'entité suivante. Pour ce faire, elle ajoute des informations complémentaires à chaque paquet y compris des codes identifiant le programme d'application émetteur et récepteur ainsi qu'une somme de contrôle (ou *checksum).* L'entité TCP réceptrice utilise cette somme de contrôle pour vérifier si le paquet reçu est intact, et utilise le code de destination pour identifier le programme d'application auquel le paquet devrait être livré.

Le modèle TCP/IP recherchait, avec le protocole IP, l'interconnexion des réseaux, il va essayer, avec le protocole TCP, de satisfaire les exigences de résistance aux incidents et de disponibilité en cas d'encombrement. Ce protocole TCP offre un service de communication fiable entre processus appartenant à des ordinateurs reliés à des sous-réseaux éventuellement distincts. Ce service est de type circuit virtuel, c'est-à-dire orienté connexion.

2.1.3.3.4 Couche application:

Au niveau le plus élevé, les utilisateurs invoquent des programmes d'application qui accèdent à l'Internet. Une application interagit avec le ou les protocoles de niveau transport pour transmettre ou recevoir des données. La couche application peut également accéder directement à IP. Chaque programme d'application choisit sa propre forme de données, qui peut être une séquence de messages ou un flot de bytes. Quelque soit la forme, la couche application passe les données à la couche transport pour la livraison.

La figure ci-dessous illustre la structure générale de TCP/IP:



Les protocoles ARP/RARP, IP, ICMP, UDP et TCP seront détaillés dans les chapitres suivants.

D'autres protocoles utilisent également le protocole IP pour transporter des informations relatives à la gestion de l'Internet. Il s'agit des protocoles GGP *(Gateway to Gateway Protocol), RIP (Routing Information Protocol), EGP (Exterior Gateway Protocol).* Tous ces protocoles aident les passerelles dans leur fonction de routage des datagrammes, la plupart d'entre eux seront seront développés dans ce texte.

Chaque ordinateur hôte qui veut utiliser un réseau interconnecté basé sur le modèle TCP/IP doit disposer d'un module IP et d'un module TCP. L'unité de données échangée entre deux modules TCP s'appelle le "segment TCP".

Pour pouvoir offrir une communication fiable entre deux ordinateurs, le protocole TCP utilise les mécanismes suivants:

- L'acquittement positif et la retransmission.
- Le contrôle de flux.
- Le séquencement.
- Le multiplexage.
- Les connexions.
- Les priorités et la sécurité.

Tous ces mécanismes sont généralement connus sauf peut être la technique de multiplexage. Celle-ci est totalement différente de celle utilisée dans la terminologie OSI. Le multiplexage TCP/IP permet à plusieurs processus d'un même ordinateur de communiquer simultanément par le même module TCP. Pour y parvenir, le protocole TCP fournit un ensemble de ports dans chaque ordinateur. Ces ports permettent au module TCP d'identifier les processus et d'associer les données, reçues ou à envoyer, à un processus. Un numéro de port concaténé à l'adresse internet forme un *socket.* Une paire de *sockets* identifie une connexion.

La plupart des systèmes d'exploitation fournissent des accès synchrones aux ports. Les paquets reçus sont mis dans une file d'attente particulière d'un port. Ces paquets seront traités ultérieurement. De même, le système d'exploitation bloque les processus qui tentent d'extraire les données d'un port avant qu'un message arrive.

Pour communiquer avec un autre port, un émetteur doit connaître l'adresse Internet de la machine de destination et le numéro de port de destination du processus. Avec chaque message, l'émetteur fournit un numéro de port de la machine de destination à laquelle le message est destiné ainsi qu'un numéro de port de la machine source à laquelle les réponses doivent être adressées. Ainsi, il est possible pour chaque processus recevant un message de répondre à son émetteur.

Deux machines doivent s'accorder sur les numéros de ports avant de commencer les échanges. Par exemple, quand une machine α veut obtenir un fichier d'une machine β, elle a besoin de connaître le numéro de port utilisé par le programme

de transfert de fichiers sur la machine β. Il y a deux approches fondamentales pour l'affectation des ports:

- La **première** approche consiste à donner à une autorité centrale le droit d'assigner les numéros de ports nécessaires et de publier la liste de ces numéros. Les logiciels seront ainsi construits en respectant cette liste. Cette approche est appelée affectation universelle et les numéros de port assignés spécifiés par l'autorité sont appelés les affectations de port connues. (Par exemple numéro de port FTP : 20et 21)

- La **deuxième** approche de l'affectation des ports utilise des liaisons dynamiques. Dans cette approche les ports ne sont pas connus globalement. Chaque programme qui a besoin d'un port se voit affecté un numéro de port par le logiciel du réseau. Pour connaître l'affectation du port courant sur une autre machine, il est nécessaire d'envoyer une demande de la forme: comment atteindre le service de transfert de fichiers. La machine cible répond en donnant le numéro de port correct qui doit être utilisé.

Les concepteurs Internet ont adoptés une approche hybride qui affecte à quelques numéros de ports une priorité, ceux-ci sont disponibles pour des sites locaux ou pour les programmes d'application. Les numéros de ports affectés ont leurs 8 bits d'ordre bas mis à zéro. Ainsi, seulement 255 ports peuvent être affectés. Ces numéros de ports sont les mêmes pour TCP et UDP (cfr. Tableau à la fin du chapitre consacré à TCP).

Dans le modèle TCP/IP, on a aussi voulu que certains protocoles d'application puissent encore disposer d'un service de type datagramme au niveau correspondant à la couche de transport. Ceci explique la présence du protocole UDP qui, par rapport au protocole IP, n'ajoute que la fonction de multiplexage.

Les protocoles TCP et UDP ne présentent aucun intérêt s'ils ne sont pas utilisés par des logiciels d'application. Le modèle TCP/IP a choisi d'associer à chaque application un protocole particulier qui utilisera un service de transport fourni, soit par TCP, soit par UDP. Ainsi, pour l'application de transfert de fichiers, on retrouve, au niveau correspondant à la couche application, les protocoles FTP et TFTP. Le premier utilise le protocole TCP tandis que le second utilise le protocole UDP. Cette couche comprend également un protocole de terminal virtuel, Telnet qui utilise TCP, un protocole de messagerie, SMTP *(Simple Mail Transfer Protocol)* qui utilise TCP, un protocole de serveur de nom, NSP *(Name Server Protocol)* qui utilise UDP et bien d'autres protocoles.

Dans ce chapitre, nous avons présenté le modèle TCP/IP. Ce modèle, composé de quatre couches, s'articule autour des protocoles IP et TCP. Le protocole IP vise essentiellement l'interconnexion des réseaux tandis que le protocole TCP cherche à offrir un service de communication fiable. Dans ce qui suit, nous allons présenter la démarche suivie par l'ISO ainsi qu'une comparaison des deux modèles.

2.1.3.4 Architecture en couches du modèle OSI

Tout comme le modèle TCP/IP, le modèle OSI essaie d'apporter une solution aux problèmes que pose l'interconnexion de réseaux. Le problème principale est l'interconnexion des réseaux ayant des protocoles, des services, des interfaces, et des formats de paquets différents. D'autres problèmes sont reliés à l'organisation des sessions, à la protection ou à la référence des fichiers, etc.

Pour remédier à ces problèmes, TCP/IP et OSI proposent des architectures hiérarchiques qui résultent des efforts de plusieurs constructeurs.

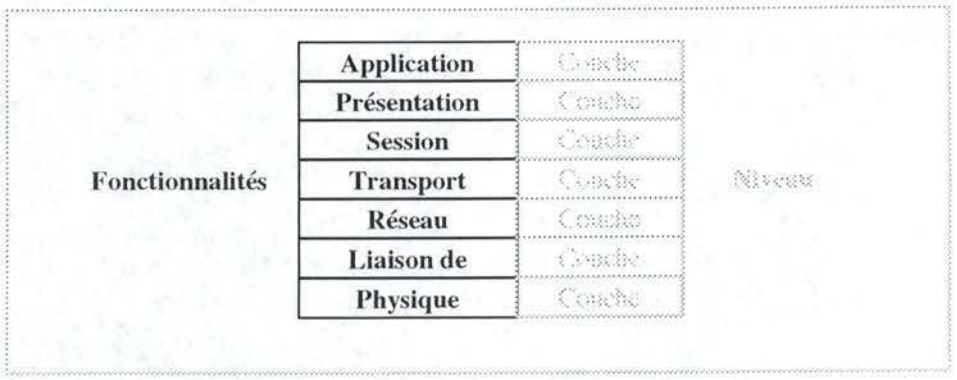
Dans OSI, le problème a été abordé de manière *top-down, en* partant d'une description d'un haut niveau d'abstraction qui impose peu de contraintes, et procédant à des descriptions de plus en plus raffinées, elles intègrent des contraintes de plus en plus strictes. Dans le monde OSI, on reconnaît **trois niveau d'abstraction**: l'architecture OSI, les spécifications de service et les spécifications de protocole.

- **L'architecture OSI** est spécifiée très scrupuleusement par la norme ISO 7498 et ses annexes. Elles se composent de deux parties : la première décrit les éléments de cette architecture tandis que la seconde énumère les services et les fonctions des sept couches choisies par l'ISO. L'idée de base de la division en couches est que chaque couche ajoute de la valeur aux services fournis par l'ensemble des couches inférieurs de telle sorte que la couche la plus haute dispose de l'ensemble complet des services nécessaires pour exécuter les applications.

- Le **deuxième aspect** est occupé par des spécifications des services OSI. Ce niveau permet de mettre en évidence la distinction qui existe entre utilisateurs et fournisseurs de services.

- Le **troisième aspect** est celui des spécifications des protocoles OSI. Chaque spécification de protocole définit très précisément quelle information de contrôle doit être envoyée et quelles procédures doivent être utilisées pour interpréter cette information de contrôle.

Cette architecture en 7 couches est illustrée dans la figure ci-dessous:



Plusieurs protocoles sont associés au modèle OSI. Ils sont ensuite adaptés par les réseaux. Ces réseaux se composent de commutateurs de paquets complexes qui contiennent l'intelligence nécessaire pour le routage des paquets. Les hôtes ne sont pas directement connectés au réseau de communication. Chacun des hôtes est relié à un commutateur de paquets utilisant une ligne de communication. La connexion entre un hôte et un commutateur de paquets est généralement un réseau minuscule composé d'une liaison en série. L'hôte doit suivre une procédure complexe pour transférer des paquets sur le réseau.

Les **sept couches** ont les spécifications suivantes:

2.1.3.4.1 Couche physique

Cette couche spécifie un standard pour l'interconnexion physique entre les ordinateurs hôtes et les commutateurs de paquets, elle spécifie également les procédures utilisées pour le transfert des paquets d'une machine à une autre. Dans le modèle de référence, le niveau 1 spécifie l'interconnexion physique ainsi que les caractéristiques du voltage et du courant électriques.

2.1.3.4.2 Couche de liaison de données

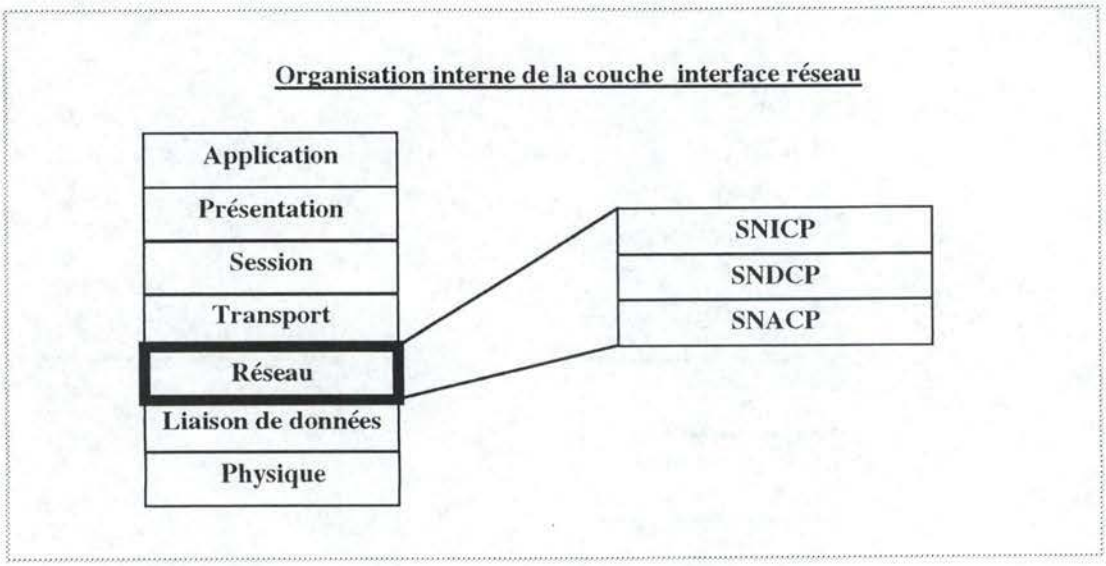
Le niveau 2 spécifie la manière dont la donnée est transmise entre un hôte et un commutateur de paquets. On utilise généralement le terme « trame » pour désigner une unité de donnée ainsi transmise. Et comme le matériel livre seulement un flot de bits, le protocole de niveau 2 doit définir le format des trames et doit spécifier la manière dont les deux machines reconnaissent les limites de ces trames. Ce protocole de niveau 2 dispose également d'un système de détection d'erreurs car les erreurs de transmission peuvent détruire des données. La transmission étant non fiable, l'échange d'accusés de réception permet aux deux machines de savoir si une trame est bien reçue.

2.1.3.4.3 Couche réseau

Le modèle ISO spécifie des fonctionnalités qui viennent compléter la définition de l'interaction entre un hôte et un réseau. Cette couche définit l'unité de transfert de base qui traverse le réseau, ainsi que les concepts d'adressage de destination et de routage. Un réseau pourrait permettre aux paquets définis par les protocoles de niveau 3 d'être plus large que la taille des trames qui peuvent être transmis au niveau 2. Le logiciel du niveau 3 assemble un paquet dans la forme exigée par le réseau et utilise le niveau 2 pour l'envoyer, éventuellement en pièces, au commutateur de paquets. Le niveau 3 doit également régler les problèmes d'encombrements du réseau.

D'une façon plus générale, la structure proposée par l'ISO pour cette couche, représentée dans la figure ci-dessous, se compose de trois sous-couches. Au niveau le plus bas, le protocole SNACP *(SubNetwork ACces Protocol)* assure les fonctions de réseau qui sont spécifiques à un sous-réseau réel particulier, et qui concernent en particulier le routage dans le sous réseau. Cette sous-couche ne fournit pas toujours le

service désiré. Pour faire face à cette situation et pour adapter ce protocole SNACP, deux sous-couches supplémentaires ont été ajoutées. Au niveau le plus élevé, la fonction d'interconnexion est prise en charge par un protocole SNICP *(SubNetwork Independent Convergence Protocol)* qui fournit à la couche de transport un service de réseau OSI, et qui exploite un jeu bien défini de fonctions offertes par un protocole intermédiaire appelé SNDCP *(SubNetwork Dependent Convergence Protocol).* Le protocole SNICP est donc indépendant des caractéristiques du sous-réseau réel. le protocole SNDCP sert essentiellement à assurer un découplage entre les caractéristiques qui sont liées au sous-réseau et l'organisation du protocole qui fournit le service de réseau à la couche transport.



2.1.3.4.4 Couche de transport

Le niveau 4 fournit un transfert de données fiable de bout en bout entre un l'hôte de destination et l'hôte source. Cette fiabilité est différente de celle offerte par les basses couches des protocoles. Celles-ci offrent des vérifications de fiabilité au niveau réseau à chaque transfert.

2.1.3.4.5 Couche session

Cette couche s'occupe de la gestion des données qu'elle reçoit de la couche du dessus ou du dessous. Elle fournit des services de transfert, de contrôle et de gestion des données sur une connexion de sessions. Ceci améliore le service fiable de transfert de données de bout en bout offert par la couche de transport d'au-dessous.

2.1.3.4.6 Couche présentation

Cette couche s'occupe de la représentation des données. Elle fournit une représentation commune pour toutes les informations des applications quand elles transitent entre les processus d'application paires. La couche se compose de fonctions nécessaires aux programmes d'application utilisant le réseau. Les routines standards

qui permettent de compresser les textes ou de convertir les images graphiques en séquences de bits pour leur transmission sur le réseau est un exempts typique des fonctions de cette couche.

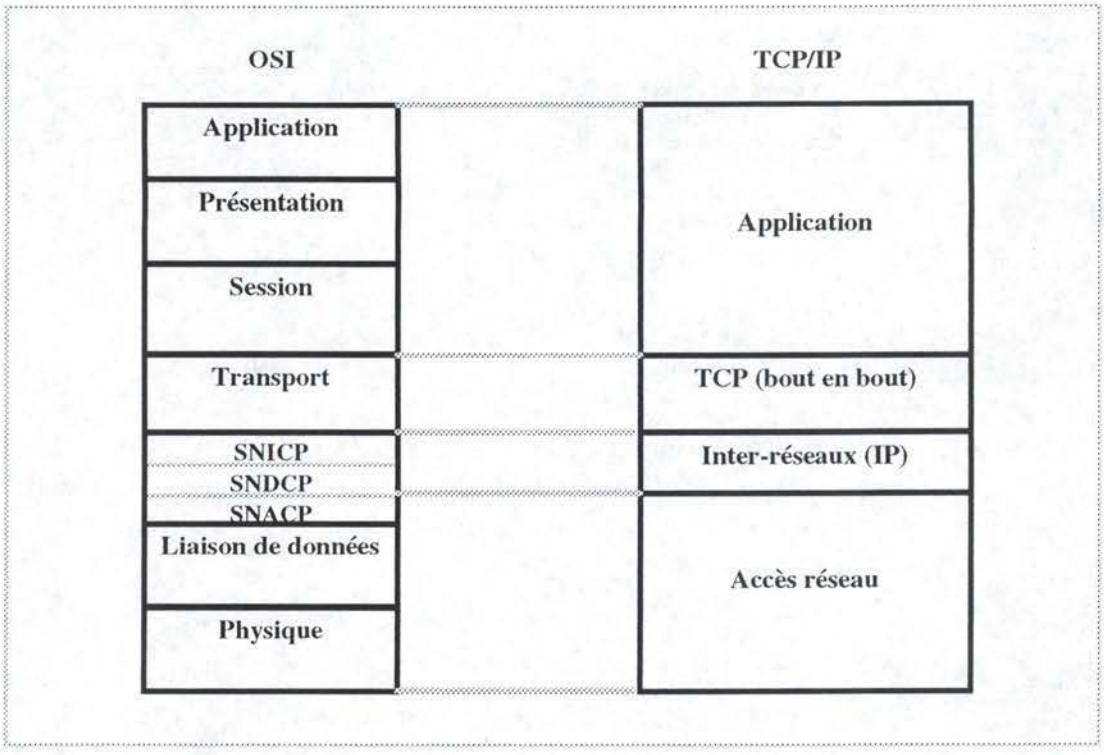
2.1.3.4.7 Couche application

Finalement, cette couche s'occupe des programmes application qui utilisent le réseau. Le courrier électronique ou les programmes de transfert de fichiers sont les exemples les plus connus.

2.1.3.5 TCP/IP versus OSI

2.1.3.5.1 Introduction

Au niveau de leurs architectures, la comparaison va s'articuler autour des quatre couches du modèle TCP/IP. Avant de passer en revue chacune de ces couches, nous rappelons au tableau ci-dessous la correspondance généralement admise entre les couches des deux modèles .



Le but poursuivi par OSI est de permettre à n'importe quel ordinateur situé n'importe où dans le monde de communiquer avec un autre ordinateur tant que ceux-ci respectent les normes OSI. Ceci entraîne de la part de l'ISO, du CCITT et de l'ECMA une grande activité dans la définition de normes dans les couches 1,2 et 3. A l'opposé, le modèle TCP/IP peut utiliser presque tous les réseaux disponibles sur le marché en respectant les objectifs définis par le DoD. Ceci explique l'intérêt relatif manifesté par le DoD au niveau accès réseau. A ce niveau, nous pouvons qualifier la

philosophie OSI de restrictive dans la mesure où les systèmes doivent respecter les normes OSI tandis que la philosophie DoD est résolument libérale car le modèle DoD vise à l'interconnexion de tout système commercialisé.

Les principaux services offerts par la couche inter-réseaux du modèle DoD et la couche réseau du modèle OSI varient d'un modèle à l'autre. Le niveau inter-réseau offre avec le protocole IP un service datagramme unique tandis que la couche réseau d'OSI fournit un service datagramme (sans connexion) analogue à celui d'IP et un service circuit virtuel (avec connexion). Le service IP du modèle DoD a opté pour un service datagramme.

Ce service permet:

- l'utilisation des passerelles pour éviter la gestion des informations d'états concernant chaque circuit créé,
- la poursuite des les communications bout à bout par des chemins alternatifs dans le cas où une passerelle ou un réseau viendrait de tomber en panne,
- et une très grande marge de manoeuvre pour définir de nouveaux protocoles autres que TCP et UDP au niveau bout en bout.

En ce qui concerne la fonction d'interconnexion des réseaux, le modèle DoD propose une solution unique: l'interconnexion des réseaux au moyen de passerelles implémentant le protocole IP. Le modèle OSI propose, quant à lui, de nombreuses solutions, mais celles-ci ne sont pas encore toutes complètement spécifiées. Le solution proposée par l'ISO est fort analogue à celle du protocole IP, c'est pourquoi il est également appelé protocole CL-IP *(ConnectionLess - Internet Protocol)* ou ISO-IP *(International Standards Organization Internet Protocol).* Ce protocole est typiquement un protocole SNICP. On peut également remarquer que le protocole IP du DoD recouvre les fonctions des protocoles SNDCP et SNICP dans la mesure où il doit également harmoniser les services offerts par les différents réseaux.

Le niveau bout en bout du DoD ainsi que la couche transport de l'ISO offrent deux types de service transport: un service transport orienté connexion et un autre orienté "sans connexion". Ce dernier est prévu pour les applications temps réel portant sur l'acquisition et la distribution de données. Le service orienté connexion est bien adapté à la plupart des autres applications informatiques; il est le plus utilisé.

Pour assurer ces deux services, le modèle DoD a spécifié deux protocoles: le protocole TCP pour offrir le service circuit virtuel et le protocole UDP pour le service datagramme. Cette simplicité du modèle DoD repose sur le fait que le niveau inter-réseaux offre un service unique de transport non fiable.

Le modèle OSI, pour sa part, en a défini quatre pour offrir un service de transport orienté connexion et un pour le service sans connexion. Dans ce modèle, la couche transport est confrontée à une diversité de services offerts par la couche réseau.

2.1.3.5.2 Principales différences entre les deux modèles

Il y a deux différences importantes entre le protocole des couches Internet et OSI. La première différence tourne autour de la fiabilité, alors que la deuxième concerne la localisation de l'intelligence dans tout le système.

2.1.3.5.2.1 Bout à Bout contre Fiabilité niveau liaison.

Une différence majeure entre les protocoles TCP/IP et les protocoles OSI est liée au contrôle d'erreurs. Dans le modèle OSI, ce contrôle se fait entre les différents niveaux. Cependant, dans le modèle TCP/IP le contrôle se fait de bout en bout.

En général, dans le modèle OSI, le logiciel du protocole détecte et manipule les erreurs à tous les niveaux. Au niveau liaison de données, des protocoles complexes garantissent un transfert correct entre un hôte et le commutateur de paquets avec lequel il est connecté. Les sommes de contrôle *(checksum)* accompagnent chaque paquet de donnée transféré, et le récepteur accuse réception de chaque paquet reçu. Le protocole de niveau liaison comprend un délai d'attente *(timeout)* et des algorithmes de retransmission qui empêchent la perte de données et fournit un redressement automatique après une panne éventuelle du matériel. Tous les niveaux fournissent, ainsi, une fiabilité de la transmission entre les machines.

Au niveau 3, par exemple, du modèle OSI un système de détection d'erreurs et de récupération des paquets transférés sur le réseau est offert, il utilise les sommes de contrôle ainsi que les techniques du délai d'attente et de retransmission. Au niveau 4, un service de fiabilité de bout en bout est également offert, il a une correspondance entre la source et la destination pour vérifier la livraison.

Internet fonde ses couches de protocoles sur l'idée que la fiabilité est un problème de bout en bout. La philosophie de l'architecture est simple: construire l'Internet de façon à ce qu'il peut manipuler le chargement prévu, mais autorise des liaisons ou des machines individuelles de perdre des données ou de les corrompre sans essayer de les récupérer de façon systématique.

En fait, il n'y a aucune couche de liaison qui s'occupe de la fiabilité dans la plupart des logiciels Internet. C'est plutôt la couche transport qui est chargée de la détection des erreurs et du problème de récupération. Le niveau liaison, libéré de ces vérifications, rend le logiciel d'Internet plus organisé et facile à implémenter. Les passerelles intermédiaires peuvent écarter des datagrammes corrompus dus aux erreurs de transmission, elles peuvent écarter tous les datagrammes qui ne peuvent pas être livrés. Elles peuvent également écarter les datagrammes quand le taux d'arrivées dépasse la capacité de la machine, en redirigeant ces datagrammes vers d'autres chemins pour les retarder et ceci sans informer la source ni la destination. Les liaisons non fiables ont pour conséquence que quelques datagrammes peuvent ne pas arriver à destination. La détection et la récupération des datagrammes perdus est réalisé entre la source et la dernière destination et pour cela elle est appelée une vérification de bout en bout. Le logiciel qui s'occupe de cette fiabilité de bout en bout

se trouve dans la couche transport, il utilise les sommes de contrôle, les accusés de réception, et les délais d'attente pour contrôler la transmission. Ainsi à la différence du modèle ISO, le logiciel du protocole Internet concentre le contrôle de fiabilité dans une seule couche.

2.1.3.5.2.2 contrôle

Une autre différence entre le modèle OSI et le modèle Internet apparaît quand on veut situer le lieu exact de l'autorité et du contrôle.

En règle général, les principaux réseaux du modèle OSI sont considérés comme étant des utilitaires qui fournissent un service de transport. Les vendeurs qui offrent le service contrôlent l'accès au réseau et le trafic des moniteurs pour l'enregistrement dans la comptabilité et pour la facturation. Le vendeur de réseaux s'occupe des problèmes tels que: le routage, le contrôle de flux, les accusés de réception internes et la fiabilité des transferts. Le réseau est un système complexe et indépendant par lequel on peut atteindre des ordinateurs hôtes; les hôtes participent eux-mêmes très peu dans les opérations des réseaux. De plus, si l'on considère l'interconnexion dans le modèle OSI, on est très proche de ce qui se fait dans TCP/IP.

Internet, cependant, exige des hôtes de participer dans presque tous les protocoles du réseau. Comme on l'a déjà mentionné ci-dessus, les hôtes implémentent de façon active les détections d'erreurs de bout en bout et leurs corrections. Ils participent également dans le routage puisqu'ils doivent choisir une passerelle quand ils envoient des datagrammes, ils participent aussi dans le contrôle du réseau puisqu'ils doivent manipuler les messages de contrôles ICMP. Ainsi, le réseau Internet peut être vu comme un système de livraison de paquets relativement simple lié à des hôtes intelligents.

2.1.3.5.2.3 Migration vers le standard OSI

Etant donné que la migration vers le standard OSI est un but avoué (si pas réel) pour tout le monde, il paraît aussi évident que cette migration ne se fera pas du jour au lendemain sans problème. Il faudra donc dans un premier temps composer avec les solutions existante de la meilleure manière possible. C'est-à-dire qu'il faudra composer avec le standard de fait qu'est TCP/IP avant de pouvoir migrer totalement vers le standard OSI issu d'un travail sérieux de modélisation.

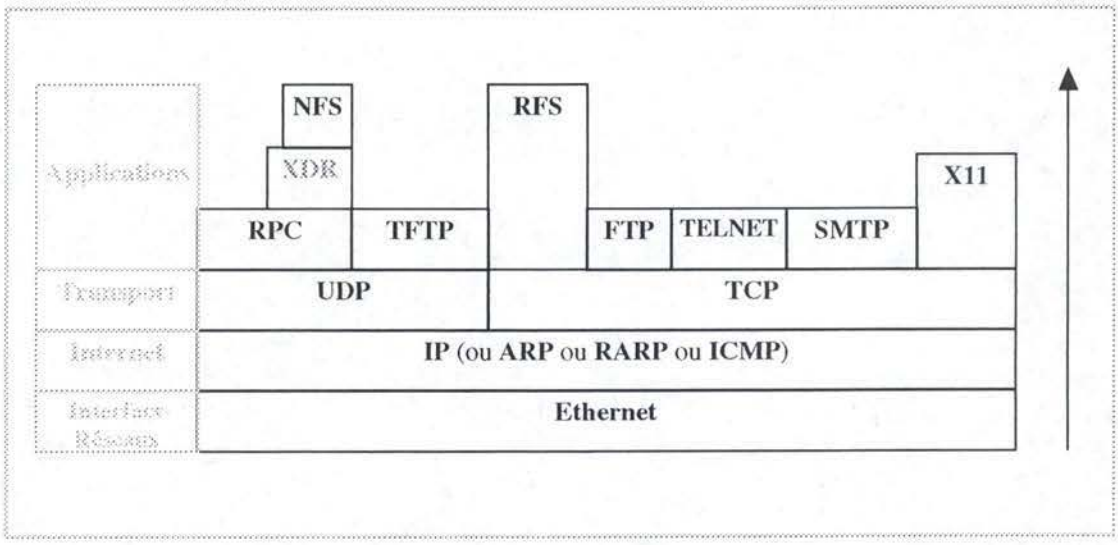
Le standard OSI a rencontré et continue de rencontrer certaines réticences de la part des utilisateurs et ceci est entre autre dû au fait qu'il existe une solution bien établie, qui a déjà pu être rodée et qui maintenant a fait preuve de fiabilité. C'est la solution TCP/IP. Les utilisateurs se méfient des solutions nouvelles et préfèrent partir sur de bonnes bases en faisant bon emploi de l'expérience acquise. Ceci dit, on peut sans trop de risques faire confiance à la fiabilité des protocoles des couches basses de la pile OSI. Mais on sait que le modèle OSI propose souvent plusieurs variantes de protocoles et qu'il n'y a pas encore accord sur le choix de ces protocoles. On pense ici à la dissension entre les modes connectés et les modes non connectés par exemple. Il se fait que les standards OSI pour le mode non connecté n'ont pas encore atteint un degré satisfaisant de maturité. Voilà donc une première raison de se tourner vers la solution TCP/IP, du moins temporairement, pour ceux qui ont plus confiance dans le

mode non connecté. Il s'agit de la confiance en une solution qui est le fruit d'une dizaine d'années d'expérience et qui, de plus, est répandue; face à la méfiance en une solution nouvelle dont l'attrait est surtout la cohérence théorique (grâce à un découpage logique en couches) mais qui n'a pas encore pu faire ses preuves sur le plan pratique.

Le but de tous les architectes de réseaux est de trouver une architecture telle que le maximum d'hôtes (et donc de personnes) puissent y être connectés. Ils veulent aussi pouvoir utiliser le maximum de produits proposés sur le marché. C'est bien pour cette raison que la nécessité d'un standard ne se discute plus mais c'est aussi pour cette raison qu'un architecte de réseau doit faire des choix stratégiques pour que son réseau rencontre ces exigences le plus rapidement possible.

Dans le même ordre d'idée, on dit qu'une des faiblesses d'OSI, ce sont ses standards et le temps qu'il faut pour les établir. Quand on a des décisions à prendre pour un réseau, on ne peut pas toujours attendre que le standard ait été approuvé par toute la communauté et ait été de plus implémenté par toute la communauté. Une deuxième raison de se tourner vers la solution TCP/IP vient donc du défaut de lenteur de développement des produits OSI et surtout la lourdeur de ses différents protocoles.

Dans ce qui suit, les principaux protocoles de suite TCP/IP et les protocoles satellites vont être présenté dans l'ordre croissant des numéros de couche du modèle Internet. Ils vont en fait être présenté selon le sens de la flèche dans le schéma suivant.



2.2 Protocole de niveau interface réseau : Ethernet.

2.2.1 Introduction

Un réseau local est composé d'ordinateurs personnels, de postes de travail, de périphériques et de logiciels, le tout relié par des câbles, des cartes de communication et des programmes de gestion de réseau. Ethernet comprend uniquement les logiciels

et les supports de communication; il permet la livraison de données grâce à la technique de diffusion (*broadcast*) et utilise les protocoles CSMA/CD. Il est composé d'un câble coaxial passif qui permet d'interconnecter tous les composants actifs.

2.2.2 Les différentes topologies

Il existe plusieurs topologies: le bus, l'étoile, l'anneau etc. Le choix de l'une ou de l'autre sera déterminé par la vitesse à laquelle on veut travailler, par la disposition des lieux, par le type de câble que l'on désire utiliser ou qui est déjà installé, par les applications, etc.

En fait, tous ces facteurs sont interdépendants car dans certains cas plusieurs types de topologies peuvent être employés simultanément. La topologie de type bus est probablement la plus répandue à l'heure actuelle. Il s'agit d'une structure simple où chaque machine est reliée à un seul et même câble. Nous appellerons "noeud" chaque connexion avec le câble.

2.2.3 Les protocoles de base

Ethernet supporte donc une topologie en bus et utilise au niveau matériel le protocole à contention avec détection de collisions dit CSMA-CD *(Carrier Sense, Multiple Access with Collision Detection).*

Une topologie de branchement en bus ne permet pas de boucles; c'est-à-dire, qu'il n'y a qu'un seul chemin pour la transmission des paquets entre deux hôtes sur un même réseau.

Le protocole **CSMA-CD** est analogue à une réunion autour d'une table ronde où chacun des interlocuteurs (ordinateurs) peut prendre la parole; pour la prendre il attend d'abord une accalmie dans la conversation (aucun trafic sur le câble du réseau). Si deux interlocuteurs commencent à parler en même temps *(collision),* ils arrêtent la communication, écoutent sur le réseau, attendent un peu, puis l'un d'entre eux recommence à parler.

Le délai réel d'attente pendant une détection de collision *(Collision Detection)* est presque aléatoire, évitant ainsi le scénario suivant:

- transmission simultané sur le réseau,
- détection de la collision,
- attente du même intervalle de temps,
- et retransmission à nouveau synchrone, c-à-d collision.

L'algorithme employé pour le calcul des délais est appelé « *Truncated binary exponentiel backoff* ». Si le nombre de collisions successives pour une même trame atteint 16, la transmission est abandonnée.

2.2.4 La connexion et l'expansion des réseaux Ethernet

Dans une topologie en bus, les extrémités sont "bouchées" par des petites pièces métalliques appelées terminateurs. Ces terminateurs ne sont, en fait, rien d'autre que des impédances de valeurs identiques à l'impédance caractéristique du câble.

Lorsqu'il faut agrandir le réseau, il suffit d'enlever un de ces "bouchons", de rajouter la nouvelle station de travail et de remettre le terminateur après celle-ci. Bien sûr, comme pour la plupart des réseaux, plus le nombre de stations connectées augmente, plus le temps de réponse augmente. La limite théorique est de 255 stations, mais cela reste bien sûr théorique car un réseau comportant un tel nombre de postes aurait une vitesse assez faible. Nous verrons lors des attaques qu'il est important de surveiller les connexions afin d'empêcher des branchements illicites directement sur le réseau.

2.2.5 Les routeurs et passerelles (*gateways*)

- Les **routeurs** sont des ordinateurs qui contiennent deux ou plusieurs interfaces Ethernet et qui fonctionnent comme un agent de circulation dans un carrefour encombré.

  Un ordinateur principal est souvent utilisé comme routeur pour des raisons économiques. On utilise les serveurs des fichiers comme routeurs pour des réseaux ayant des stations de travail sans disques puisque la grande partie du trafic dans le réseau local est entre le serveur et ses clients; ainsi le chargement supplémentaire laissé chez le serveur fonctionnant comme un routeur n'est plus important.

- Les **passerelles** sont des machines hôtes qui font la conversion des protocoles, elles s'occupent également du routage. Ainsi, une machine qui parle à la fois Ethernet et X.25 et qui passe des paquets entre ces deux réseaux sera appelé passerelle. Les routeurs sont souvent appelés passerelles, surtout s'ils sont des ordinateurs principaux très chargés.

  Un autre type de passerelle est celui du courrier qui doit faire la conversion nécessaire pour permettre aux messages destinés à un hôte, par exemple, IBM BITNET d'arriver sur un Intemet TCP/IP et être reroutés et remballés en une forme comprise par le système BITNET.

2.2.6 Les normes

Ethernet est normalisé sous la norme IEEE 802 (ISO 8802) qui est consacrée aux réseaux locaux (802.4, Bus à jeton; 802.5, anneau à jeton; 802.3, méthode d'accès CSMA/CA OU CSMA/CD).

2.3 Protocoles de niveau Internet (niveau réseau).

2.3.1 Protocole IP

2.3.1.1 Introduction

Comme nous l'avons vu, l'Internet TCP/IP est constitué de trois ensembles de services superposés sous forme de couches dépendantes les unes des autres. En bas de la structure, on trouve la couche IP. Celle-ci fournit un service de livraison non fiable et sans connexion *(connectionless).* Dans la couche de dessus, on trouve le service TCP qui est fiable et qui fournit une plate forme de haut niveau dont dépendent les programmes d'applications. Ceux-ci constituent la couche la plus élevé.

Le logiciel d'Internet est conçu autour de ces trois couches conceptuelles, qui sont arrangées hiérarchiquement, son succès résulte de cette architecture robuste et adaptable. Cette architecture présente des possibilités de remplacement de certains services sans toucher au reste.

Le protocole qui définit la non fiabilité et le mécanisme de livraison sans ouverture de connexion est appelé *Internet protocol,* habituellement référencé par ses initiales : IP. Le protocole définit l'unité de base de transfert de données à travers le réseau TCP/IP, il spécifie dès lors le format exact des données qui vont être transmises. Le logiciel du protocole IP exécute une fonction de routage pour choisir un chemin par lequel les données doivent être envoyées.

En plus des spécifications des formats de données et du routage, il dispose d'un ensemble de règles qui expliquent la non fiabilité de la livraison des paquets. Ces règles définissent, la manière selon laquelle les hôtes et les passerelles devraient traiter les paquets, comment et quand les messages d'erreurs devraient être générés, et quelles sont les conditions dans lesquelles les paquets devraient être écartés.

Ce service est appelé *connectionless* car chaque paquet est traité indépendamment de tous les autres, il est donc envoyé sans ouverture de connexion. Les paquets envoyés d'une machine à une autre peuvent passer par différents chemins sur lesquelles quelques uns d'entre eux peuvent se perdre. Le logiciel d'Internet tente de livrer les paquets en faisant de son mieux *(best-effort delivery)* et donc il n'écarte jamais un paquet délibérément; le problème de non fiabilité est causé généralement par l'épuisement des ressources ou par une panne dans le réseau.

2.3.1.2 Structure statique : le datagramme

2.3.1.2.1 Introduction

L'unité de transfert de données de base est le datagramme d'Internet ou tout simplement datagramme. Comme pour les trames du réseau physique, un datagramme est divisé en deux parties: l'en-tête *(header)* et la zone de données *(data area).* L'en-tête, comme pour la trame, contient les adresses de la machine source et de destination et un champ pour spécifier le type de l'information du datagramme. Mais la différence avec une trame est que l'en-tête d'un datagramme contient les adresses IP cependant l'en-tête de la trame contient l'adresse physique.

2.3.1.2.2 Adresses IP

2.3.1.2.2.1 Introduction

Internet est un grand réseau virtuel bâti sur l'interconnexion de sous-réseaux physiques au moyen de passerelles. Il est conçu et implémenté entièrement sous forme d'un logiciel. Les concepteurs ont donc eu toute la liberté de choisir les formats et les tailles de paquets, les adresses, les techniques de livraison, etc. Rien n'a été imposé par le matériel. Pour les adresses, ils ont choisi un plan analogue à celui des réseaux physiques dans lesquels on attribue à chaque hôte une adresse sous forme d'un entier. Cet entier s'appelle l'adresse Internet.

A chaque hôte sur le réseau Internet est affecté une adresse Internet unique d'une longueur de 32 bits qui sera utilisée dans toutes les communications. Cet identificateur est universel. Les bits d'une adresse Internet de tous les hôtes sur un sous-réseau donné ont un préfixe commun.

2.3.1.2.2.2 Les adresses Internet

Chaque adresse est représenté par une paire *(netid, hostid), où netid* est l'identifiant du réseau, et *hostid* l'identifiant de l'hôte sur ce réseau. En pratique, les adresses Internet se présentent principalement sous trois formes, comme l'illustre le tableau ci-dessous:

Classe	Bits d'ordre le plus élevé	NetID (bits)	HostID (bits)	Nombre d'adresses
A	0	7	24	16 777 214
B	10	14	16	65 534
C	110	21	8	254
D	1110	Multicast group		268 435 456
E	1111	Utilisation expérimentale		-

Les adresses de la classe A sont utilisées pour des réseaux qui ont plus de 216 (65636) hôtes, 7 bits sont consacrés au *netid* et 24 au *hostid.* Cependant, les adresses de la classe B qui sont utilisées pour des réseaux de taille moyenne c'est-à-dire entre 28 (256) et 216 hôtes, allouent 14 bits au *netid* et 16 bits au *hostid.* Finalement, les adresses de la classe C qui ont moins de 28 hôtes, allouent 22 bits au *netid* et seulement 8 bits au *hostid.*

Cette structure des adresses Internet permet l'extraction des zones *netid et hostid* en un temps constant . Les passerelles, qui sont à la base du routage à partir de *netid,* dépendent de l'efficacité de cette extraction.

Une adresse Internet ne peut pas toujours être le seul identifiant d'un hôte, car les passerelles qui relient deux réseaux physiques ont nécessairement deux adresses Internet. La plupart des machines ont plus d'une connexion physique. Ces machines exigent plusieurs adresses Internet; comme ces adresses spécifient un réseau et un hôte sur ce réseau, elles n'identifient pas un hôte mais une connexion à un réseau. Ainsi, une passerelle qui connecte n réseaux possède n adresses Internet différentes, une pour chaque connexion réseau.

Le codage des informations du réseau en adresses Internet n'est pas le seul moyen de rendre le routage efficace, ces adresses Internet peuvent également spécifier des réseaux.

Une adresse Internet dont tous les bits de son *hostid* mis à zéro est utilisé pour faire référence à un réseau. Un hôte ayant ce type d'adresse ferait référence à lui même. Et si le champ *netid* d'une adresse Internet est constitué de zéros, l'hôte ferait référence à ce même réseau.

Un autre avantage de cet adressage est qu'il comprend une adresse de diffusion qui fait référence à tous les hôtes sur le réseau. D'après les standards, si les bits du champ *hostid* d'une adresse sont mis à 1, celle-ci est réservée à la diffusion. Sur Ethernet la diffusion peut être aussi efficace que la transmission normale, sur d'autre technologies elle est assurée par les modules du réseau et est plus lente qu'une simple transmission. La diffusion peut être absente sur certains réseaux.

Quand une machine reçoit un paquet avec le champ *netid* mis à zéro et le champ *hostid* égale à l'identifiant d'hôte, ce paquet est interprété comme étant transmis du même réseau sur lequel les deux machines sont attachés et à travers lequel le paquet est arrivé.

On utilise cette technique dans le cas où on ne connaît pas l'adresse réseau avec lequel on veut communiquer. L'hôte utilise le *netid* zéro temporairement pour pouvoir recevoir des réponses des autres hôtes avec l'adresse complète du réseau et ainsi il pourrait l'enregistrer et l'utiliser pour une prochaine émission.

Les adresses Internet se composent de quatre entiers séparés par des points, où chaque entier donne la valeur d'un octet de l'adresse Internet. Pour les réseaux de la classe A, le premier entier varie de 0 à 128, pour la classe B, il se situe entre 128 et 192[4], et pour la classe C, il est de 192 à 256.

Tous les adresses Internet sont assignées par une autorité centrale qui s'appelle le NIC *(Network Infor-pnation Center)*.. Le NIC attribue des numéros à la partie réseau de l'adresse Internet et délègue la responsabilité de l'attribution de numéros d'hôtes à l'organisme demandeur.

[4] C'est le cas des facultés : 138.48.x.y

Il attribue aux réseaux locaux, tel que Ethernet, les numéros de la classe C car la plupart de ces réseaux ne dépassent pas les 255 hôtes. Aux grands réseaux, tel que ARPANET, sont attribué les numéros de la classe A.

Le NIC s'occupe seulement de l'attribution d'adresses Internet aux réseaux reliés ou qui seront reliés à l'Internet DARPA. Plusieurs autres groupes qui utilisent les protocoles TCP/IP attribuent eux-mêmes les adresses réseaux car ils n'ont pas besoin de s'interconnecter avec Internet.

2.3.1.2.2.3 Faiblesse de l'adressage Internet

Le codage des adresses Internet a quelques désavantages :

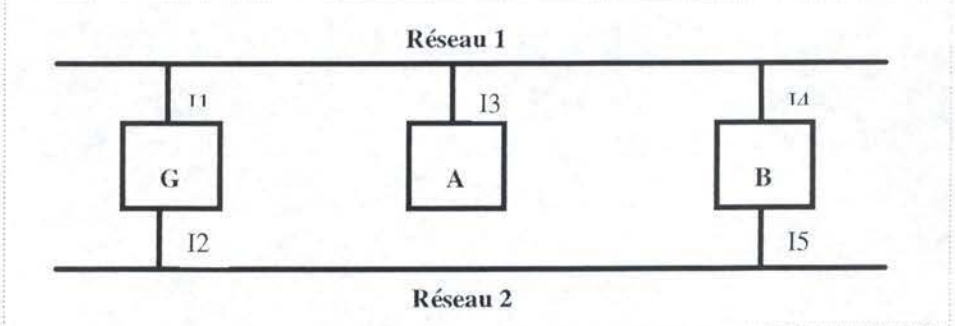
- Le plus évident est que les adresses dépendent des connexions et non des hôtes. Si on déplace un hôte d'un réseau à un autre, son adresse Internet devrait changer.

- Une autre faiblesse de ce système d'adressage est que si un réseau de classe C dépasse 255 hôtes, les adresses Internet de ces derniers doivent être changé au système d'adressage de la classe B. Ce problème paraît assez simple, mais en réalité il est difficile de corriger les adresses d'un réseau et cela prend beaucoup de temps. Et comme la majorité des logiciels ne peuvent pas manipuler plusieurs types d'adresses pour le même réseau physique, les administrateurs ne peuvent agir de façon progressive pour introduire les nouvelles adresses. Ils doivent arrêter complètement l'utilisation des anciennes adresses du réseau et commencer l'utilisation des nouvelles dès que la correction est faite.

Comme on l'a déjà indiqué, le routage sera basé sur ces adresses Internet et particulièrement sur l'identifiant du réseau pour pouvoir prendre des décisions de routage. Considérons un hôte ayant deux connexions au réseau Internet. On sait qu'un tel hôte dispose de plus d'une adresse Internet. Ainsi, le chemin pris par les paquets pendant la transmission dépend de l'adresse utilisée.

Connaissant une seule adresse d'un hôte de destination, il peut être parfois insuffisant car il pourrait être impossible de l'atteindre en utilisant cette adresse.

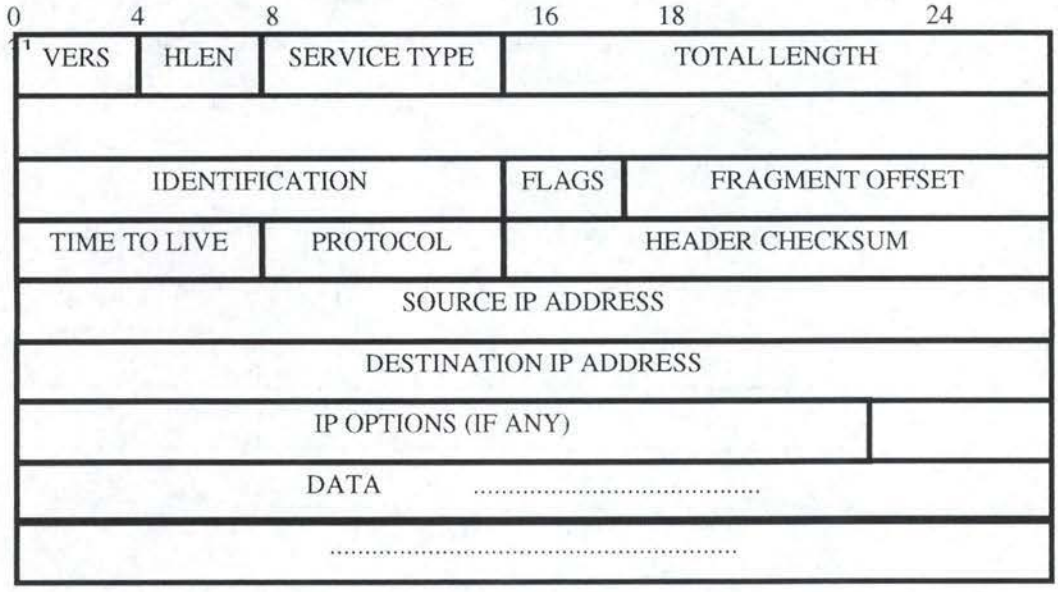
Considérons l'exemple suivant illustré ci-dessous:



Les deux hôtes A et B peuvent communiquer directement en utilisant le réseau 1. Ainsi, A s'adresse à B en utilisant l'adresse Internet I4. Un second chemin existe entre A et B à travers la passerelle G et est utilisée quand A envoie des paquets à l'adresse Internet 15. Si la connexion de B sur le réseau 1 échoue, et que la machine reste active, A ne pourrait plus atteindre B avec l'adresse 14. Pour pouvoir l'atteindre il faudrait spécifier l'adresse 15. Donc, si l'interface 14 se déconnecte, A devrait utiliser I5 pour atteindre B, en routant les paquets à travers la passerelle G. Ainsi, chaque hôte devrait connaître toutes les adresses des autres car en cas de panne d'une connexion il peut saisir une autre adresse encore disponible.

2.3.1.2.3 Le datagramme

Comme le traitement des datagrammes se fait par le logiciel d'Internet correspondant à la couche IP, le contenu et le format du datagramme ne dépendent plus de la machine. Ce format est le suivant:



2.3.1.2.4 Formats et contenus des champs

2.3.1.2.4.1 **VERS:**

Ce champ de 4 bits contient la version du protocole IP utilisé pendant la création du datagramme. On utilise ce champ pour que les machines émettrices et réceptrices et les passerelles employées se mettent d'accord sur le format du datagramme. Tous les logiciels IP doivent vérifier le champ VERS avant de traiter un datagramme.

Si les standards changent, les machines rejetteront les datagrammes dont la version du protocole diffère de la leur.

2.3.1.2.4.2 **HLEN:**

Ce champ est également d'une longueur de 4 bits, il donne la longueur de l'en-tête du datagramme en mots de 32 bits. Tous les champs de l'en-tête ont la même longueur excepté le champ IP OPTION et le champ correspondant PADDING.

Un en-tête qui mesure 20 octets n'a pas les champs IP OPTION et PADDING, son champ HLEN est donc égale à 5.

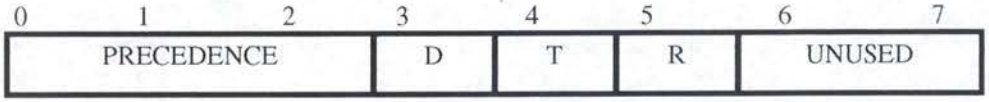
2.3.1.2.4.3 **TOTAL LENGTH:**

Ce champ donne la longueur en octets de tout le datagramme IP *(header + data)*. La taille de la zone de données peut donc être calculée en retirant HLEN du TOTAL LENGTH. Comme le champ TOTAL LENGTH a une longueur de 16 bits, la taille maximale possible d'un datagramme IP est de 65535 octets. Dans la plupart des applications cette limitation est acceptable. Cette limite pourrait devenir importante si dans l'avenir il y avait des réseaux à grande vitesse dont les paquets de transmission de données dépassent 65535 octets.

2.3.1.2.4.4 SERVICE TYPE :

Ce champ a une longueur de 8 bits, il spécifie la façon dont le datagramme doit être manipulé.

Il est divisé en cinq parties :



2.3.1.2.4.4.1 ***PRECEDENCE:***

Ce sous-champ a une longueur de 3 bits, il spécifie la priorité du datagramme, sa valeur varie de 0 (priorité normale) à 7 (commandes réseaux), permettant ainsi à l'émetteur d'indiquer l'importance de chaque datagramme.

Malgré que la majorité des hôtes et des passerelles ignorent ce champ, ce concept peu être utilisé puisqu'il offre un mécanisme de contrôle de l'information en mettant des priorités sur les données. Par exemple, si tous les hôtes et passerelles utilisent ce champ de priorité, il serait possible d'implémenter des algorithmes capables de contrôler l'encombrement des réseaux. Cet encombrement n'affecte pas ces algorithmes.

2.3.1.2.4.4.2 ***D, T et R:***

Chacun de ces sous-champs a une longueur d'un bit. Ils spécifient le type de transport désiré par le datagramme. Les spécifications du type de transport n'est donc qu'une suggestion ou une indication pour l'algorithme de routage, qui va l'aider à choisir le chemin optimal. Ces algorithmes se basent sur leurs connaissances des technologies et du matériel disponibles sur ces chemins.

Un réseau Internet ne garantit donc pas le type de transport demandé car il pourrait ne pas exister un chemin vers la destination qui possède ces propriétés.

Quand les trois bits sont mis à 1 :

- pour le champ D, la durée de transmission du datagramme doit être brève *(low delay);*
- *pour* T, le débit de transmission doit être très élevé;
- et enfin pour R, une grande fiabilité du support est exigée.

Si une passerelle connaît plus d'un chemin pour atteindre une destination, elle **pourrait** utiliser le champ type de transport pour sélectionner une route dont les caractéristiques sont les plus proches de celles demandées.

Trois champs de l'en-tête du datagramme s'occupent **du contrôle, de la fragmentation** et **de l'assemblage :** l'IDENTIFICATION, le FLAG, et le FRAGMENT OFFSET.

2.3.1.2.4.5 **IDENTIFICATION:**

Ce champ contient un entier unique qui identifie le datagramme et qui sert, par exemple, en cas de fragmentation. Comme les passerelles, par exemple, peuvent fragmenter un datagramme, elles copient la majorité des champs de l'en-tête du datagramme dans chaque fragment. Le champ IDENTIFICATION doit donc être copié sur chaque fragment, pour permettre à la destination de savoir si le fragment appartient ou non à un datagramme donné. Quand le fragment arrive, la destination utilise le champ IDENTIFICATION avec l'adresse source du datagramme pour identifier le datagramme.

Les ordinateurs qui envoient des datagrammes doivent gérer une valeur unique du champ IDENTIFICATION pour chaque datagramme. Une technique utilisée par le logiciel IP est de tenir un compteur en mémoire, qui est incrémenté chaque fois qu'un nouveau datagramme est créé, et assigne le résultat au champ IDENTIFICATION.

2.3.1.2.4.6 **FRAGMENT OFFSET:**

Les grands paquets sont généralement fragmentés à l'émission et rassemblés à l'arrivée. Les fragments obtenus ont exactement le même format qu'un datagramme complet. Pour un fragment, le champ FRAGMENT OFFSET indique l'emplacement de ce fragment dans le datagramme initial, mesuré en unités de 8 octets, et commençant par l'emplacement numéro zéro. Pour ré-assembler le datagramme, la destination doit recevoir tous les fragments commençant par le fragment qui possède le numéro d'emplacement zéro jusqu'au fragment ayant le numéro le plus élevé. Les fragments n'arrivent pas nécessairement dans l'ordre, et il n'y a aucune communication entre la passerelle qui fragmente le datagramme et la destination qui essaye de les ré-assembler. Si un des fragments se perd, le datagramme entier doit être écarté.

2.3.1.2.4.7 **FLAGS:**

Ce champ est composé de 3 bits ; les deux bits d'ordre le plus faible s'occupent du contrôle de la fragmentation. Le premier de ces deux bits indique si le datagramme

peut être fragmenté ou pas, il est appelé "le bit de non fragmentation" (DO NOT FRAGMENT), s'il est mis à 1, le datagramme ne doit pas être fragmenté. Les premiers bits de droite (d'ordre faible) du FLAG indiquent le dernier fragment (le fragment qui a le plus grand OFFSET). Comme le champ LENGTH de l'en-tête du fragment s'applique à la taille du fragment et non à celle du datagramme initial, la destination ne peut pas utiliser cette longueur pour vérifier si elle a collecté tous les fragments. Ainsi elle a besoin du bit "encore des fragments" (MORE FRAGMENT) qui indique si l'entièreté du datagramme initial a été reçue.

2.3.1.2.4.8 TIME TO LIVE:

Ce champ indique le temps de vie maximal, en secondes, d'un datagramme dans le réseau Internet. En principe, l'idée est à la fois simple et importante: chaque fois qu'un hôte transmet un datagramme, il fixe un temps maximal durant lequel un datagramme peut survivre. Chaque passerelle, tout le long du chemin entre la source et la destination, traite le datagramme. Elle vérifie le temps restant et décrémente ce champ de 1, si *le TIME TO LIVE* atteint le zéro, le datagramme est écarté.

Ainsi, les datagrammes ne peuvent plus circuler indéfiniment dans le réseau même si la table de routage est corrompue.

Il est difficile d'estimer le temps exact car les passerelles ne connaissent pas habituellement le temps de transit dans le réseau. Pour simplifier le traitement, les hôtes et les passerelles supposent que chaque transfert de réseau dure une unité de temps. Ainsi, ils doivent décrémenter le champ TIME TO LIVE d'une unité chaque fois qu'ils traitent un en-tête.

2.3.1.2.4.9 PROTOCOL:

Ce champ est analogue à celui du type de la trame d'Ethernet. Les protocoles de haut niveau utilisent PROTO pour indiquer le format et le contenu de la donnée en identifiant le type de protocole. L'organisation entre le protocole de haut niveau et la valeur entière utilisée dans ce champ doit être gérée par une autorité centrale pour garantir la standardisation à travers tout le réseau Internet.

2.3.1.2.4.10 HEADER CHECKSUM :

Ce champ assure l'intégrité de la valeur de l'en-tête. La somme de contrôle IP ou IP CHECKSUM est formé en traitant l'en-tête comme une séquence d'entiers de 16 bits. Ces entiers sont additionnés en utilisant le complément à un (*one's complement*) arithmétique, et ensuite on prend le complément à un du résultat. Pour des besoins de calcul de la somme de contrôle, ce champ est supposé contenir zéro.

La somme de contrôle se trouve dans l'en-tête séparée des données. L'en-tête occupe d'habitude moins d'octets que les données, la séparation des sommes de contrôle réduit le temps de traitement dans les passerelles qui ont seulement besoin de calculer les sommes de contrôle des en-têtes. La séparation permet également au protocoles de haut niveau de choisir leur propre formules pour la somme de contrôle.

En plus, les protocoles de haut niveau sont forcés d'ajouter leur propre somme de contrôle sinon, ils risquent de voir des données corrompues passer sans être détectées.

2.3.1.2.4.11 SOURCE IP ADDRESS et DESTINATION IP ADDRESS:

Ces champs contiennent des adresses Internet de 32 bits, respectivement de l'émetteur du datagramme et du récepteur.

2.3.1.2.4.12 DATA:

Ce champ contient les données du datagramme. La longueur de ce champ dépend de ce qui va être transmis dans le datagramme.

2.3.1.2.4.13 IP OPTION / PADDING:

Le champ IP OPTION, discuté ci-dessus, a une longueur variable. Le champ PADDING dépend de l'option sélectionnée, il représente les bits contenant zéro dont on peut avoir besoin pour assurer que l'en-tête Internet s'étend à un multiple exact de 32 bits. La longueur du champ de l'en-tête étant spécifiée en unités de mots de 32 bits.

2.3.1.2.4.13.1 Les options du datagramme d'Internet:

Le champ IP OPTION qui suit le champ adresse de destination n'est pas exigé dans chaque datagramme; les options sont inclues généralement pour les tests ou la mise au point des réseaux. La longueur de ce champ varie suivant les options choisies.

Quelques options ont une longueur d'un octet, d'autres ont des longueurs variables. Quand il y a des options présentes dans un datagramme, elles apparaissent comme étant contiguës, elles ne sont pas séparées par des séparateurs spéciaux. Chaque option consiste en un code option d'un octet, une longueur d'un octet, et un ensemble d'octets de données pour l'option. L'octet du code d'option est divisé en trois champs comme la figure ci-dessous le montre :

Les champs consistent en un drapeau d'une longueur d'un bit appelé COPY qui contrôle le traitement des options par les passerelles durant la fragmentation, quand ce bit est à 1, il indique que l'option devrait être copiée dans tous les fragments, s'il est à 0, l'option est seulement copiée dans le premier fragment; d'un deuxième champ de 2 bits appelé OPTION CLASS, et d'un troisième champ de 5 bits appelé OPTION NUMBER, ces deux champs indiquent la classe générale de l'option et donnent une option spécifique dans cette classe.

Le tableau ci-dessous montre comment ces classes sont assignées:

Option Class	Explications
0	contrôle de datagramme ou de réseau
1	réservé pour des utilisations futures
2	mesures et mises au point
3	réservé pour des utilisations futures

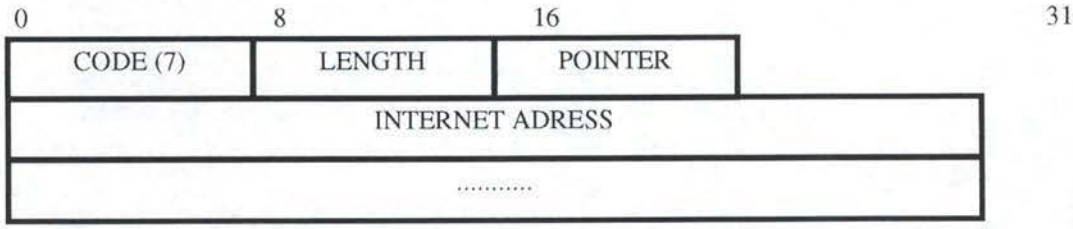
Les options possibles qui peuvent accompagner un datagramme IP sont données dans la figure ci-dessous, pour chaque OPTION CLASS on donne sa valeur OPTION NUMBER. On remarque que la majorité des options sont utilisées pour des buts de contrôle:

OPTION CLASS	OPTION NUMBER	LENGTH	DESCRIPTION
0	0	1	Fin de la liste des options. Utilisé si les options ne se terminent pas à la fin du datagramme.
0	1	1	Pas d'opérations.
0	2	11	Des restrictions de manipulations et de sécurité.
0	3	var	Routage libre à partir de la source (LOOSE SOURCE ROUTING). Utilisée pour router le datagramme à travers un chemin spécifié.
0	7	var	Enregistrement des routes. Utilisée pour tracer une route.
0 d'un flux	8	4	Identificateur de flux. Utilisée pour transporter un identificateur SATNET.
0	9	var	Routage stricte à partir de la source (STRICT SOURCE ROUTING). Utilisée pour router les datagrammes à travers un chemin spécifique.
2 Utilisée	4	var	Internet TIMESTAMP. pour enregistrer les marques de temps le long du chemin.

2.3.1.2.4.13.2 Les options d'enregistrement des routes:

Les options de routage et de marque de temps sont les plus importants car elles offrent un moyen de contrôler comment les passerelles routent les datagrammes. Cette option permet à la source de créer une liste vide d'adresses IP et s'arrange pour que chaque passerelle qui manipule le datagramme ajoute son nom à la liste.

Le format de l'option enregistrement de la route est présenté ci-dessous:



Comme décrit ci-dessus, le champ CODE contient le numéro et la classe d'option (7 pour l'enregistrement de la route). Le champ LENGTH indique la longueur totale de l'option comme elle apparîcit dans le datagramme IP, les trois premiers octets compris.

Le champ INTERNET ADDRESS dénote la zone réservé pour les adresses Internet, et un champ POINTER qui donne le complément dans l'option du prochain emplacement disponible.

Chaque machine qui manipule le datagramme ajoute son adresse à la liste d'enregistrement des routes. Pour cela, la machine compare d'abord le pointeur et le champ longueur. Si le pointeur est plus grand que la longueur, la liste est alors pleine, et la machine expédie le datagramme sans rien insérer dans la liste. Si la liste n'est pas pleine, la machine insère son adresse de 4 octets et incrémente le pointeur de 4 unités.

2.3.1.2.4.13.3 Les options de routage à partir de la source (IP SOURCE ROUTING)

Une autre option que les constructeurs de réseaux trouvent intéressante est l'option routage à partir de la source (SOURCE ROUTING). L'idée est de fournir un moyen pour l'émetteur de dicter un chemin à travers Internet.

Par exemple, pour tester le débit d'un réseau donné, les administrateurs du système peuvent s'arranger pour forcer les datagrammes IP sur un de ces réseaux en utilisant le routage à partir de la source.

En pratique, IP supporte deux formes de routage à partir de la source. La première forme appelé le routage stricte de la source (STRICT SOURCE ROUTING), qui comprend une séquence d'adresses Internet. C'est à dire que les adresses spécifient le chemin que le datagramme doit suivre pour atteindre sa destination. Le chemin entre deux adresses successives dans la liste doit consister en un seul réseau physique; une erreur survient si une passerelle ne peut pas suivre le routage stricte de la source. L'autre forme, appelé routage non strict à partir de la source (LOOSE SOURCE ROUTING), comprend également une séquence d'adresses Internet. Elle indique au datagramme de suivre la séquence des adresses Internet, mais permet le saut de plusieurs réseaux entre les adresses de la liste.

Tout au long du chemin, les passerelles doivent mettre leurs adresses du réseau local dans la liste. Ceci est exigé par les deux options de routage. Ainsi, quand le datagramme arrive à sa destination, il contient une liste de tous les adresses visitées, exactement comme la liste produite par l'option enregistrement des routes.

Le format de l'option routage à partir de la source est comme celle de l'enregistrement des routes. Chaque passerelle examine les champs du pointeur et de la longueur pour voir si la liste a été épuisée. Si oui, le pointeur sera plus grand que la longueur; la passerelle route le datagramme vers sa destination comme d'habitude. Si la liste n'est pas épuisé, la passerelle suit le pointeur, prend l'adresse Internet, la remplace avec l'adresse de la passerelle, et route le datagramme en utilisant l'adresse obtenue de la liste.

2.3.1.2.4.13.4 L'option TIMESTAMP

Cette option fonctionne comme celle de l'enregistrement des routes, cependant, elle contient une liste initialement vide et chaque passerelle visité tout le long du chemin entre la source et la destination va remplir un article de cette liste. Les entrées sont cependant constituées par l'heure et la date à laquelle la passerelle a manipulé le datagramme, exprimés en millisecondes à partir de minuit (GMT).

Ce TIMESTAMP a également une option qui permet à la source d'exiger de chaque passerelle qu'elle insère à la fois un TIMESTAMP et son adresse Internet dans la liste. D'habitude, cette option est utilisée pour contrôler la performance d'un réseau. Avec l'enregistrement des routes et les TIMESTAMP, le récepteur peut savoir exactement quels sont les chemins que le datagramme a empruntés.

2.3.1.3 Fonctionnement dynamique

2.3.1.3.1 Fragmentation

Le réseau transporte les datagrammes d'une machine à l'autre. Pour que ce transport soit efficace, chaque datagramme doit être transmis dans une trame physique distincte, ainsi l'intégrité du datagramme est garantie. L'idée de transporter un datagramme dans une trame est appelé encapsulation.

Le réseau ne reconnaît pas le format du datagramme et ne comprend pas l'adresse IP de destination; alors quand une machine envoie un datagramme IP à une autre, tout le datagramme sera mis dans la partie donnée de la trame. Dans le meilleur des cas, le datagramme IP s'adapte bien à la taille de la trame, ce qui rend la transmission efficace. Pour cela, la sélection de la taille maximale d'un datagramme IP doit bien correspondre à celle de la trame.

La quantité maximale de données qui doit être transférée dans une trame, est fixée par la technologie de commutation utilisée; par exemple pour Ethernet, la limite est de 1500 octets de données, alors que pour d'autres supports comme proNET-10 par exemple, elle peut atteindre 2044 octets par trame.

Ces unités de données s'appellent des MTU (*Maximum Transfer Unit*)ou unités maximales de transfert du réseau. Les MTU peuvent être très petites, de l'ordre de 128 octets ou moins pour certaines autres technologies. Si on adapte la taille des datagrammes aux plus petites MTU possibles, le transfert deviendrait inefficace, surtout si ces datagrammes empruntent des réseaux qui peuvent acheminer des trames beaucoup plus larges. Cependant, si on utilise des datagrammes plus larges que le minimum des MTU dans un réseau Internet, le datagramme ne s'adapterait pas toujours à une trame.

Le logiciel de TCP/IP a choisit une taille optimale du datagramme, il s'est également arrangé pour diviser les larges datagrammes en plus petites parties. Ces petites parties sont appelées fragments et le processus de cette division est appelé fragmentation.

Ces fragmentations se produisent habituellement dans les passerelles entre la source et la destination. La passerelle reçoit des paquets dans des MTU de grande taille et doit les router sur un autre réseau dans lequel les MTU sont de taille plus petites. Ces datagrammes sont fragmentés et la taille des fragments résultants doit être multiple de 8 octets. En choisissant le multiple de 8 octets le plus proche de la taille du MTU, la division du datagramme en fragments de même taille n'est plus garantie; le dernier fragment est souvent plus petit que les autres. Les fragments doivent être rassemblés pour reproduire une copie complète du datagramme originale avant d'être traité par la destination. La fragmentation et l'assemblage s'effectuent automatiquement sans que l'émetteur prenne des dispositions précises.

Un datagramme ayant, par exemple, 1400 octets de données, est divisé en trois fragments de 620 octets, chaque fragment contient un en-tête qui reprend la majorité des informations contenues dans l'en-tête du datagramme initial (excepté le bit du champ "FLAG" qui indique que c'est un fragment), suivi par autant de données que le fragment peut transporter; la longueur totale ne doit pas dépasser celle du MTU du réseau.

Dans le réseau TCP/IP, une fois que le datagramme a été fragmenté, les fragments se propagent comme des datagrammes séparés jusqu'à leur destination où ils doivent être assemblés.

La fragmentation peut générer des problèmes. D'abord les fragments peuvent traverser des réseaux différents qui peuvent avoir des MTU de taille très large ce qui diminue l'exploitation optimale des réseaux; en plus si un des fragments est perdu, le datagramme ne peut plus être assemblé. En fait, la machine réceptrice déclenche une horloge d'assemblage quand elle reçoit le premier fragment, si l'intervalle de temps critique est dépassé avant que tous les autres fragments arrivent, la machine réceptrice écarte les fragments reçus sans traiter le datagramme. Ainsi, la probabilité de perte de datagrammes augmente quand il y a une fragmentation car la perte d'un seul fragment implique la perte de tout le datagramme.

Malgré ces problèmes, la technique de l'assemblage à la destination fonctionne bien, elle permet ainsi aux fragments d'être routés indépendamment

et donc ils n'ont plus besoin de passerelles intermédiaires pour rassembler les fragments.

2.3.1.3.2 Routage des datagrammes

Les services Internet sont basés sur un système non fiable de livraison de paquets et sans ouverture de connexion . Ce système a comme unité de transfert de base le datagramme IP. Celui-ci est routé par des passerelles vers sa destination finale. L'algorithme de routage constitue l'aspect dynamique opérationnel de l'IP alors que le format du datagramme est l'aspect statique.

Le routage peut être difficile surtout avec des machines connectées par plusieurs réseaux. Le code de routage doit sélectionner un réseau, sur lequel le datagramme doit être envoyé.

L'idéal est que le logiciel de routage, pour sélectionner le meilleur chemin, examine la charge du réseau, la longueur du datagramme, ou le type de service spécifié dans l'en-tête de celui-ci. Mais malheureusement, la plupart de ces logiciels sont beaucoup moins sophistiqués, ils sélectionnent les routes en se basant sur des suppositions fixes des chemins les plus courts.

Le routage IP est analogue à celui des réseaux physiques, cependant il se fait à un niveau plus haut que ce dernier.

Pour bien comprendre le routage IP, revenons à l'architecture d'Internet. On sait qu'Internet est composé de plusieurs réseaux physiques interconnectés par des ordinateurs appelés des passerelles. Les hôtes quant à eux , sont connectés directement à un ou plusieurs réseaux physiques. Les hôtes et les passerelles participent au routage IP.

2.3.1.3.3 Types de routage

2.3.1.3.3.1 Introduction

Pour ce qui suit, on va distinguer les hôtes des passerelles, et on va supposer que les hôtes n'exécutent pas les fonctions des passerelles (qui consistent à transférer les paquets d'un réseau à un autre).

On peut distinguer **deux types** de routage, le routage direct et indirect:

- le **routage direct**: la transmission d'un datagramme se fait directement d'une machine à une autre. Il n'y a pas de machines intermédiaires, car les deux machines se trouvent sur le même réseau.

- le **routage indirect**: quand la destination n'est pas sur le même réseau, elle force l'émetteur de passer le datagramme à une passerelle pour la livraison.

2.3.1.3.3.2 Routage direct

Une machine sur un réseau physique donné peut envoyer directement une trame à une autre machine sur le même réseau. Pour transférer un datagramme IP, l'émetteur met le datagramme dans une trame physique, transforme l'adresse IP en une adresse physique, et utilise le réseau pour la livraison. Ainsi, la transmission d'un datagramme IP entre deux machines sur un seul réseau physique n'implique pas de passerelles; la trame qui contient le datagramme est envoyée directement à la destination.

Pour voir si la destination se trouve sur un des réseaux connecté directement à la machine, l'émetteur extrait la partie concernant le réseau de l'adresse Internet de destination (*netid*), et la compare à la partie réseau de sa propre adresse Internet. Si les deux parties coïncident, le datagramme peut être envoyé directement en utilisant l'adresse physique.

On peut ainsi voir les avantages de l'organisation des adresses Internet: les adresses de toutes les machines qui se trouvent sur le même réseau comprennent un identificateur de réseau commun, et comme l'extraction de cet identificateur peut être faite par quelques instructions, on peut tester si une machine peut être directement atteinte; ce qui est extrêmement efficace.

2.3.1.3.3.3 Routage indirect

Le routage indirect est plus difficile que le routage direct car l'émetteur doit identifier une passerelle vers laquelle on peut envoyer le datagramme. La passerelle doit également retransmettre le datagramme sur le réseau de destination.

Supposons qu'on a un réseau Internet composé de plusieurs sous-réseaux interconnectés par des passerelles. S'il y a seulement deux hôtes aux deux bouts, l'hôte émetteur va transmettre le datagramme vers la passerelle la plus proche. Cependant, comme tous ces sous-réseaux sont interconnectés, chacun d'entre eux doit avoir une passerelle qui le relie avec le reste des sous-réseaux.

Ainsi, l'hôte émetteur peut atteindre une passerelle en utilisant un seul réseau physique. Une fois que la trame atteint la passerelle, un programme va extraire le datagramme et une routine de routage va ensuite sélectionner la prochaine passerelle le long du chemin de la destination. Le datagramme est de nouveau placé dans une trame et envoyé au réseau suivant et donc vers une seconde passerelle, etc., jusqu'à ce qu'il peut être livré directement.

L'idée est donc que les passerelles forment une structure interconnectée de coopération, les datagrammes passent d'une passerelle à une autre jusqu'à ce qu'ils atteignent une passerelle qui peut les livrer directement à leur destination.

2.3.1.3.3.4 Algorithmes de routage

2.3.1.3.3.4.1 Routage sur base de tables

L'algorithme de routage habituel emploie sur chaque machine une table de routage. Cette table contient des informations sur les destinations possibles. Si chaque table de routage contenait des informations sur chaque adresse de destination possible, il serait impossible de maintenir ces tables à jour. De plus, comme le nombre de destinations possibles est grand, les machines n'auraient pas suffisamment de place pour ranger toutes les informations.

Les passerelles routent les paquets sans connaître les détails concernant les hôtes et leurs environnements locaux. Heureusement, le plan de l'adresse IP rend un tel routage possible. Les passerelles n'ont pas besoin de connaître tous les détails tant qu'ils connaissent la structure d'interconnexion des réseaux et le réseau de destination auquel un datagramme doit être transmis.

Quand un datagramme arrive à une passerelle, le logiciel IP localise l'adresse de destination Internet et extrait la partie concernant le réseau. La passerelle utilise ensuite l'identificateur du réseau pour prendre les décisions de routage. L'utilisation des adresses du réseau de destination plutôt que des adresses de l'hôte de destination rend le routage efficace et maintient une table de routage assez petite.

Typiquement, une table de routage contient des paires (N,G), où N est l'adresse du réseau Internet de destination, et G est l'adresse Internet d'une passerelle à laquelle les datagrammes doivent être envoyés, ils seront donc destinés aux réseaux N. Tous les passerelles listés dans la table de routage et utilisées par une machine M, doivent être reliées aux réseaux sur lesquels la machine M est directement connecté, rendant ainsi la liaison directe possible. Ainsi pour maintenir une taille minimum pour cette table, le logiciel de routage IP se base sur les adresses de réseaux de destination et non pas sur les adresses individuelles d'hôtes.

Cependant, le fait de choisir des routes basés seulement sur des identificateurs de réseaux de destination peut avoir plusieurs conséquences :

- **Premièrement**, dans la plupart des implémentations, ceci veut dire que tous les débuts de trafic pour un réseau donné prennent le même chemin. Comme résultat, même s'il existe plusieurs chemins, ils peuvent ne pas être utilisés simultanément. Aussi tous les types de trafic suivent le même chemin sans tenir compte des retards ou des débits des réseaux physiques.

- **Deuxièmement**, la seule passerelle qui peut communiquer avec l'hôte de destination est la dernière sur le chemin. Elle est la seule à savoir si l'hôte existe et s'il est opérationnel. Ainsi, nous avons a besoin de construire un chemin vers cette passerelle pour envoyer à la machine source le rapport sur les problèmes rencontrés.

• **Troisièmement**, puisque chaque passerelle va router le trafic de façon indépendante, le passage de ce trafic d'un hôte A vers un hôte B peut suivre des chemins totalement différents que celui du passage de B vers A. Pour cela, les passerelles doivent coopérer pour garantir que cette communication à deux voies soit toujours possible.

On sait maintenant que IP route les datagrammes en se basant sur une table de routage, cette table de routage doit être initialisée ou mise à jour chaque fois que le réseau change. Pour cela il y a des protocoles qui permettent aux passerelles de garder les routes cohérentes. Il est donc important de savoir que IP se base sur les tables pour ses décisions de routage, et qu'en changeant ces tables on change les routes suivies par les datagrammes.

Ainsi, le routage IP comporte des décisions sur les routes à prendre pour atteindre la destination. La livraison directe implique l'encapsulation du datagramme dans une trame physique, la transformation de l'adresse Internet de destination en une adresse physique par l'intermédiaire du protocole ARP *(Address Resolution Protocol), et* l'envoi de la trame sur le réseau.

Pour l'algorithme de routage Internet orienté tables, les décisions de routage se basent sur les adresses de réseau de destination plutôt que sur les adresses des hôtes de destination, gardant ainsi la table de routage petite et efficace. Le routage par défaut sert à garder une petite table, surtout pour les hôtes qui peuvent accéder à une seule passerelle.

2.3.1.3.3.4.2 Routes par défaut

Une autre technique utilisée pour garder une table de routage minime est de consolider plusieurs entrées en un seul cas appelé "cas par défaut". L'idée est d'avoir un logiciel de routage IP qui cherche d'abord le réseau de destination dans la table de routage. Si aucune route n'apparaît dans la table, les routines de routages envoient le datagramme au cas par défaut.

Le routage par défaut est surtout utile pour les sites disposant d'un petit ensemble d'adresses locales et ayant une seule connexion avec l'Internet. Par exemple, les routes par défaut sont plus efficaces dans les machines hôtes qui sont reliés à un seul réseau physique et ne pouvant atteindre qu'une seule passerelle qui conduit au reste de l'Internet.

Le mécanisme de routage se fait en deux tests: un pour le réseau local et un par défaut qui pointe vers la seul passerelle possible. Même si le site contient peu de réseaux locaux, le routage est simple puisqu'il consiste en quelques tests pour les réseaux locaux et un par défaut pour toutes les autres destinations.

2.3.1.3.3.4.3 *Les routes spécifiques aux hôtes*

Tous les routages sont basés sur les adresses de réseaux et pas sur des hôtes individuels. Malgré cela, la plupart des logiciels de routage IP fournissent des routes pour chaque hôte. Ces hôtes doivent être spécifiées comme étant des cas spéciaux.

Ayant ces routes par hôte, l'administrateur du réseau local aura plus de contrôle sur le réseau, ils peuvent également être utilisés pour le contrôle de l'accès. En mettant au point les connections de réseaux ou les tables de routage, la capacité de spécifier une route spéciale pour une machine individuelle se révèle très utile.

2.3.1.3.3.4.4 *Algorithme générale*

En tenant compte de tout ce qu'on avait dit, l'algorithme de routage IP devient:

Algorithme

Routage-du-datagramme-IP (datagramme, table-de-routage)

Extraire l'adresse IP de destination, ID, du datagramme

Calculer l'adresse IP du réseau de destination, IN

Si IN correspond à une adresse réseau directement connectées
Alors Envoyer le datagramme à sa destination sur ce réseau;

```
(Ceci implique que ID est transformé en une adresse physique, que le
     datagramme est encapsulé, et que la trame est transmise)
```

Sinon si ID apparaît comme une route de hôte-spécifique

Alors Router le datagramme comme spécifié dans la table;

Sinon si IN apparaît dans la table de routage

Alors Router le datagramme comme spécifié dans la table;

Sinon si une route par défaut a été spécifié

Alors Router le datagramme vers la passerelle par défaut ;

Sinon déclarer une erreur de routage;

2.3.1.3.3.5 Manipulation des datagrammes à l'arrivée

Les paquets reçus sont également impliqués dans le routage. Quand un datagramme IP arrive à un hôte, il est livré au logiciel IP pour être traité. Deux cas se présentent: le datagramme a atteint sa destination finale, ou bien, il doit continuer son chemin. Dans le premier cas, l'IP doit passer le datagramme à l'application de destination pour le traitement. Dans le deuxième cas, l'IP retient le datagramme et le route en utilisant l'algorithme standard.

Une machine peut avoir plusieurs connexions physiques, chacune de ces connexions a sa propre adresse Internet. Quand un datagramme arrive, l'hôte doit comparer l'adresse Internet de destination avec chacune de ses adresses de réseau. Si une d'entre elles corresponde à l'adresse de destination, il passe le datagramme à l'application de destination. S'il n'y a aucune correspondance, IP décremente le champ TIME TO LIVE de l'en-tête du datagramme, il l'écarte quand ce compte atteint zéro sinon il le re-route.

Les passerelles doivent router les datagrammes reçus car c'est leur fonction principale. Plusieurs hôtes dotés d'une multitude de points de départs agissent comme des passerelles. Les hôtes qui ne sont pas désignés comme étant des passerelles ne doivent pas router les datagrammes qu'ils reçoivent; ils doivent les écarter. Il y a **quatre raisons** pour lesquelles ces hôte doivent s'abstenir d'exécuter les fonctions des passerelles:

1. Quand un tel hôte reçoit un datagramme destiné à d'autres machines, c'est qu'il y a eu un mauvais fonctionnement dans l'adressage d'Internet ou dans le routage. Le problème peut ne pas être révélé si l'hôte prenait des décisions correctives en re-routant le datagramme.

2. Le re-routage va causer un trafic inutile dans le réseau.

3. Les erreurs simples peuvent causer un chaos. Supposons que chaque hôte reroute le trafic, ce qui va se passer si une erreur déclenche la diffusion d'un datagramme qui est destiné à un hôte H, est que chaque hôte sur le réseau qui reçoit ce datagramme, le re-route vers H, celui-ci va être criblé par un grand nombre de copies.

4. Les passerelles utilisent un protocole spécial pour signaler les erreurs; les hôtes qui ne sont pas désignés pour servir comme passerelle ne signalent pas les erreurs.

2.3.2 Protocole ICMP

2.3.2.1 Introduction

2.3.2.1.1 Objectifs

On a vu que le protocole Internet offre un service de livraison de paquets non fiable (*unreliable*) et sans connexion (*connectionless*), et que les paquets traversent les passerelles pour arriver à leur hôte de destination. Si une passerelle ne peut pas router ou livrer un datagramme, ou si elle détecte des conditions inhabituelles, comme par exemple l'encombrement du réseau, elle informe les hôtes de ces conditions pour qu'ils puissent prendre des décisions pour résoudre le problème. Le mécanisme qu'utilisent les passerelles et les hôtes pour communiquer de telles informations de contrôle ou d'erreurs est spécifié dans le protocole des messages de contrôle ICMP (pour *Internet Control Message Protocol).* Ce mécanisme est utilisé par les

passerelles pour résoudre les problèmes de report de livraison, il est également utilisé par les hôtes pour tester si des destinations peuvent être atteintes.

ICMP fournit des messages pour réduire le taux de transmission ou flux à la source, des messages de REDIRECTION qui demandent à un hôte de changer ses tables de routage et des messages de demande et de réponse d'écho que les hôtes utilisent pour déterminer si une destination peut être atteinte.

2.3.2.1.2 Problèmes

Dans les systèmes sans connexion, chaque passerelle ou hôte fonctionne d'une manière autonome, en routant et en livrant les datagrammes qu'ils reçoivent sans aucune coordination avec l'émetteur. Le système fonctionne bien quand toutes les machines fonctionnent correctement et s'accordent sur le routage, mais aucun système ne fonctionne correctement d'une façon continue. Des pannes sur les lignes de communication ou sur des processeurs peuvent se produire, Internet ne peut plus livrer les paquets quand la machine de destination est temporairement ou définitivement déconnectée du réseau ou quand le compteur TIME TO LIVE expire ou quand les passerelles intermédiaires deviennent trop encombrées et ne peuvent plus traiter le trafic.

Les messages ICMP traversent le réseau Internet dans la zone de donnée des datagrammes IP comme tous les autres trafics. Quand un message d'erreurs ICMP arrive à une destination donnée, il est pris en charge par le module IP; il ne passe, donc, pas au programme d'application causant le problème.

En résumé, ICMP permets aux passerelles et aux hôtes de transmettre des messages d'erreurs et des messages de contrôle à d'autres passerelles ou hôtes; ICMP fournit ainsi une communication entre les entités Internet de deux machines en communication.

2.3.2.2 Système de livraison

Chaque message ICMP traverse le réseau Internet dans la partie donnée du datagramme IP. Les datagrammes qui transportent les messages ICMP sont routés exactement comme ceux transportant des informations. Ainsi, même les messages d'erreurs peuvent être perdus ou écartés.

Il est important de savoir que même si les messages ICMP sont introduits et transmis en utilisant IP, ICMP n'est pas considéré comme un protocole de haut niveau, mais une partie nécessaire à IP. La raison pour laquelle on utilise IP pour livrer les messages ICMP est qu'ils peuvent traverser plusieurs réseaux physiques pour atteindre leur destination finale.

2.3.2.3 Format du message ICMP

Chaque message ICMP possède un format propre, ils commencent tous par 3 champs: un champ TYPE du message sous forme d'entier à 8 bits, un champ CODE également à 8 bits qui fournit des informations complémentaires sur le type de

message, et un champ CHECKSUM à 16 bits (ICMP utilise le même algorithme de la somme de contrôle que IP, mais cette somme de contrôle d'ICMP se compose seulement du message ICMP).

Les messages ICMP se composent toujours de l'en-tête et les premiers 64 bits de données du datagramme qui a causé l'erreur. La raison pour laquelle on donne toutes ces informations (entête du datagramme + 64 premiers bits des données) est de permettre au récepteur de déterminer de façon plus précise les protocoles utilisés et de localiser le programme d'application responsable. La majorité des protocoles de haut niveau sont conçus de façon à ce que les informations cruciales soient encodées dans les premiers 64 bits.

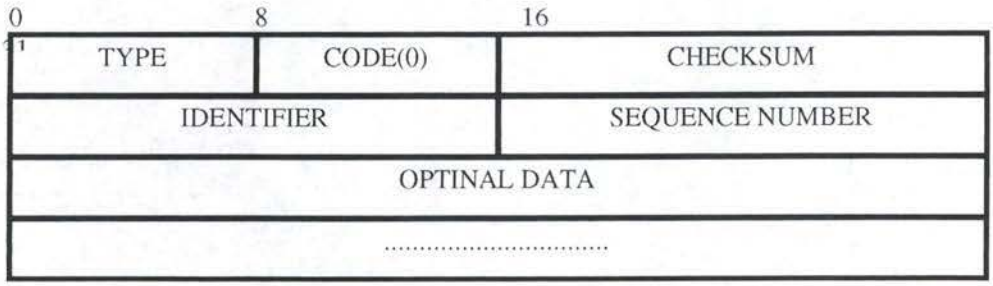
Le champ TYPE d'ICMP définit la signification du message et le format du reste du paquet. Ces types sont:

Champ type	Type de message ICMP
0	Réponse Echo
3	Destination ne peut être atteinte
4	*Source Quench*
5	Redirection (changer de route)
8	Demande Echo
11	Temps dépassé pour un datagramme
12	Problème de paramètres dans le datagramme
13	Demande d'indication de temps
14	Réponse d'indication de temps
15	Demande d'information
16	Réponse d'information
17	Demande du masque de l'adresse
18	Réponse du masque de l'adresse

2.3.2.4 Tests d'atteinte d'une destination

Un hôte ou une passerelle envoie un message ICMP de demande d'écho pour tester si une destination est vivante et si on pourrait l'atteindre.

Chaque machine qui reçoit cette requête doit formuler une réponse d'écho et la renvoie à la machine qui a envoyé la demande. Le format du message écho est illustré dans la figure ci-dessous:

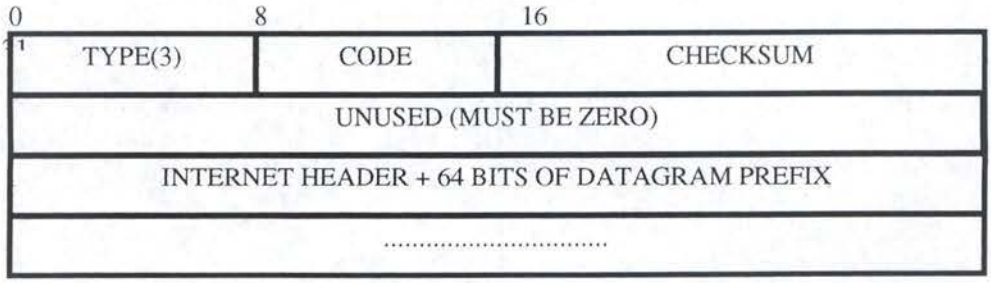


Format d'une demande ou d'une réponse d'ECHO ICMP

Le champ OPTIONAL DATA est un champ de longueur variable, il contient les données qui doivent être retournées au client. Une réponse d'écho doit retourner exactement la même donnée reçue dans la demande. Les champs IDENTIFIER et SEQUENCE NUMBER sont utilisés par le client pour faire la correspondance entre les demandes et les réponses. La valeur du champ TYPE indique si le message est une demande (8) ou une réponse (0).

Le mécanisme demande/réponse d'écho ICMP est un outil très utile dans Internet. Plusieurs systèmes offrent un programme d'application que l'utilisateur peut évoquer pour envoyer des demandes d'écho ICMP et des temps de réponse. Comme le logiciel Internet de la machine de destination manipule ICMP, il est possible d'obtenir un écho même si la machine est très chargée, c'est-à-dire même quand les processus de niveau utilisateur ne réagissent plus.

Quand une passerelle ne peut pas livrer un datagramme IP, elle retourne un message à la source pour lui indiquer que la destination ne peut pas être atteinte, en utilisant le format illustré dans la figure ci-dessous:



Format du message ICMP indiquant que la destination ne peut être atteinte (*destination unreachable*)

Le champ CODE dans le message contient un entier qui donne une description supplémentaire du problème. Les valeurs possibles de ce champ sont:

Valeur du code	Signification
0	On ne peut pas atteindre le réseau
1	On ne peut pas atteindre l'hôte
2	On ne peut pas atteindre le protocole
3	On ne peut pas atteindre l'accès
4	Fragmentation nécessaire
5	Route de la source défaillante

Une passerelle envoie des messages indiquant que le réseau ou l'hôte ne peut pas être atteint quand elle ne peut pas router ou livrer des datagrammes. On ne peut pas atteindre des destinations si le matériel est temporairement en panne, ou si

l'émetteur spécifie une adresse de destination inexistante, ou (dans des cas rares) si la passerelle n'a pas une route pour un réseau de destination.

Dans le message ICMP, le préfixe du datagramme qui a causé le problème est indiqué pour que le logiciel du protocole à la source puisse le connaître avec précision.

Si une passerelle doit fragmenter un datagramme, le bit de non fragmentation étant mis à 1, elle envoie un message de fragmentation nécessaire à la source.

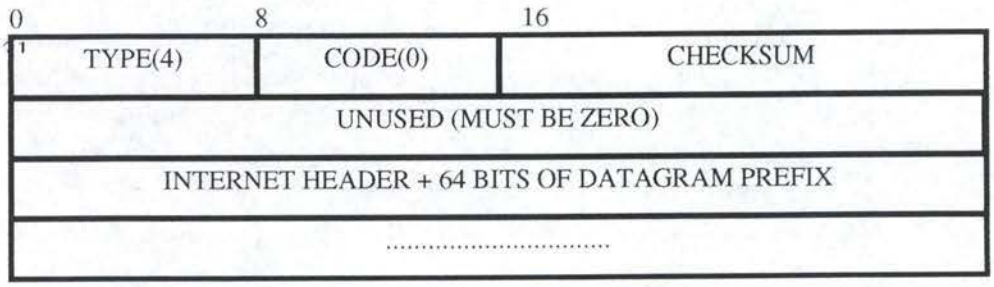
2.3.2.5 Contrôle de flux

Quand les datagrammes arrivent trop rapidement à une passerelle ou à un hôte pour le traitement, ils sont généralement écartés. Les machines qui écartent ces datagrammes envoient un message ICMP à la source pour lui demander de diminuer son taux de transmission de datagramme.

D'habitude, les machines envoient une demande à ces sources pour chaque datagramme devant être écarté. Cependant, les passerelles peuvent utiliser plusieurs algorithmes pour sélectionner ces sources.

Pour éviter l'écartement des datagrammes, les passerelles envoient ces demandes de réduction du flux seulement quand leur file d'attente devienne très longue. Il n'y a pas de message ICMP pour inverser l'effet d'une demande de réduction de flux de datagrammes. Au lieu de cela, un hôte qui reçoit des messages de ce type d'une destination D, diminue le taux de transmission des datagrammes à D jusqu'à ce qu'il ne reçoive plus de demande; il augmente, ensuite, progressivement le taux tant qu'il n'a pas reçu de nouvelles demandes.

Ces messages contiennent, en plus des champs TYPE, CODE et CHECKSUM, un champ UNUSED de 32 bits, et un champ DATAGRAM PREFIX comme illustré dans la figure ci-dessous:



Format du message ICMP de réduction du taux de transmission.

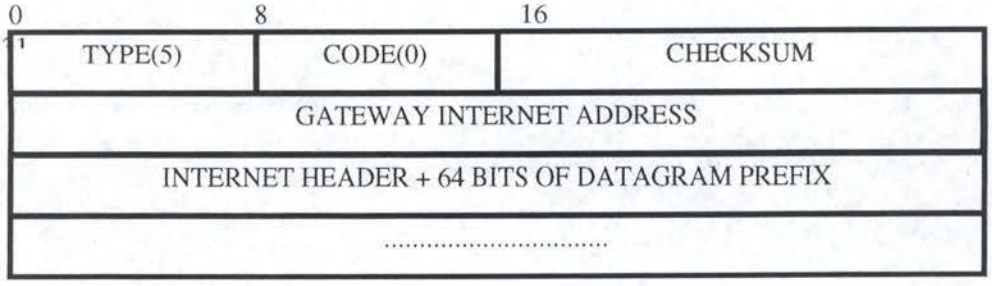
Dans tous les messages ICMP, un champ contient un préfixe du datagramme qui a déclenché la demande de réduction du taux de transmission de la source.

2.3.2.6 Changements de routes

Les tables de routage d'Internet restent d'habitude statiques pendant des périodes de temps assez longues. Les hôtes les initialisent à partir d'un fichier du disque au démarrage du système, et les administrateurs de systèmes changent rarement les routes pendant les opérations normales. Si la structure de l'interconnexion du réseau change, les tables de routage contenues dans des hôtes particuliers peuvent devenir incorrectes. Ces changements peuvent être temporaires ou permanents.

Les passerelles échangent des informations de routage périodiquement pour s'adapter à des changements de réseaux et pour mettre à jour leurs tables de routage. Ainsi, les passerelles connaissent généralement mieux les routes que les hôtes. Quand une passerelle détecte qu'un hôte est en train d'utiliser une route non optimale, elle lui envoie un message ICMP de REDIRECTION. Elle expédie également le datagramme à sa destination.

Chaque message de REDIRECTION contient, en plus des champs TYPE, CODE, et CHECKSUM, un champ GATEWAY INTERNET ADDRESS de 32 bits et un champ DATAGRAM PREFIX comme l'illustre la figure ci-dessous:



Format du message ICMP de REDIRECTION

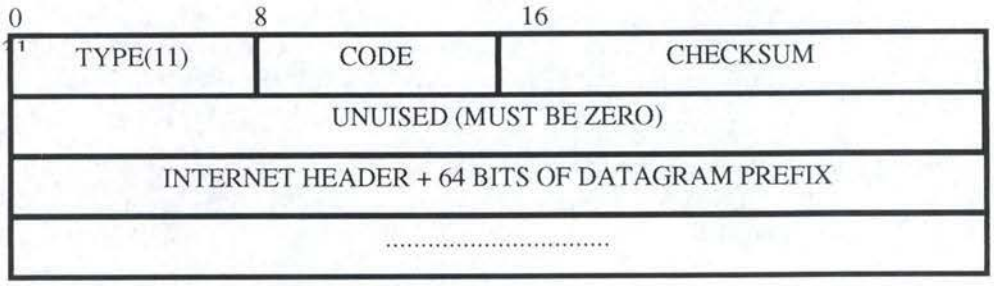
Le champ GATEWAY INTERNET ADDRESS contient l'adresse d'une passerelle que l'hôte doit utiliser pour atteindre la destination mentionnée dans l'en-tête du datagramme. Le champ INTERNET HEADER contient l'en-tête IP plus les 64 bits suivants du datagramme qui a causé le message. Ainsi, un hôte qui reçoit un message de REDIRECTION examine le préfixe du datagramme pour déterminer l'adresse de destination du datagramme. Le champ CODE d'un message de REDIRECTION spécifie davantage comment interpréter l'adresse de destination, en se basant sur les valeurs assignées suivantes:

Valeur du code	Signification
0	Rediriger les datagrammes vers le réseau
1	Rediriger les datagrammes vers l'hôte
2	Rediriger les datagrammes vers le type de service et le réseau
3	Rediriger les datagrammes vers le type de service et l'hôte

Les passerelles peuvent seulement envoyer des demandes de REDIRECTION aux hôtes sur des réseaux directement connectés et pas aux autres passerelles.

2.3.2.7 Résolution du problème de routage

Chaque datagramme IP contient un compteur TIME TO LIVE, appelé parfois un HOP COUNT. Pour prévenir que les datagrammes tournent en rond indéfiniment (PING-PONG), chaque passerelle décrémente le compte et écarte le datagramme quand il atteint zéro. Chaque fois qu'une passerelle écarte un datagramme dont le compte a atteint zéro, elle envoie un message ICMP d'excès de temps à l'émetteur du datagramme, en utilisant le format illustré dans la figure ci-dessous:



Format du message ICMP de dépassement de temps.

Le champ CODE contient une valeur spécifiant la nature du délai d'attente. Les différentes valeurs sont:

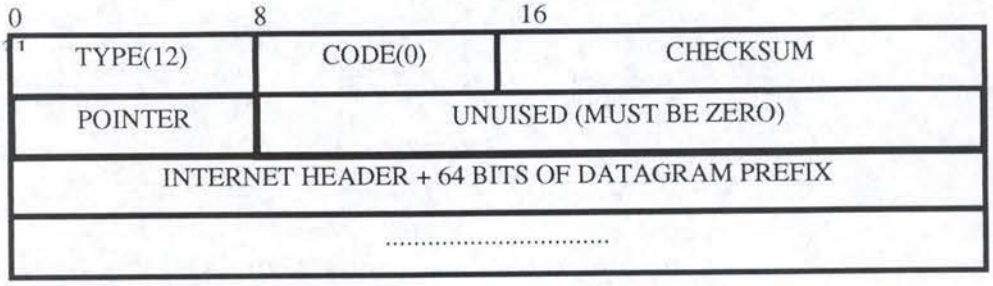
Valeur du code	Signification
0	Dépassement du compte TIME TO LIVE
1	Dépassement du délai de réassemblage des fragments

L'assemblage des fragments est une tâche qui consiste à collecter tous les fragments du datagramme. Chaque fois que le premier fragment d'un datagramme arrive, l'hôte récepteur déclenche une horloge. Si le délai du temps horloge expire avant l'arrivé de tous les pièces du datagramme, le récepteur le considère comme une erreur. La valeur 1 du champ CODE est utilisée pour signaler de telles erreurs à l'émetteur; un message est transmis pour chaque erreur de ce type.

2.3.2.8 Résolution du problème des paramètres d'en-tête

Quand une passerelle ou un hôte rencontre des problèmes avec l'en-tête du datagramme, il envoie un message indiquant un problème de paramètres à l'émetteur.

Une cause possible de tels problèmes se présente quand les arguments d'une option sont incorrects. Le message est transmis seulement quand le problème est tellement sévère que le datagramme doit être écarté. Ce format est illustré dans la figure ci-dessous:



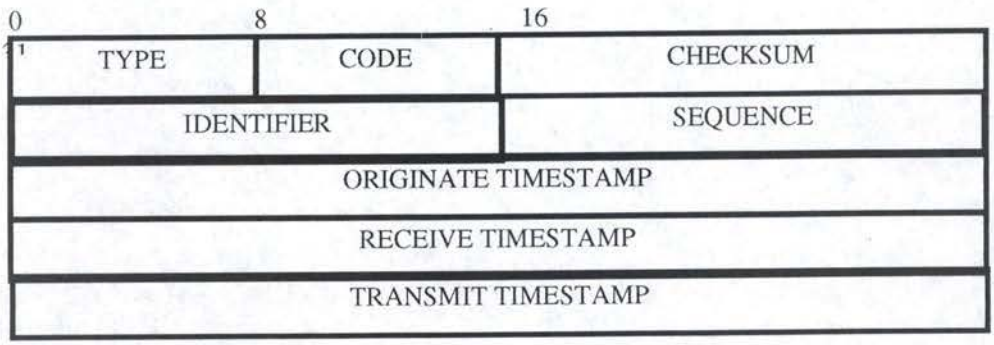
Format du message ICMP indiquant un problème de paramètres

Le champ POINTER identifie la position de l'octet, dans l'en-tête du datagramme, qui a causé le problème.

2.3.2.9 Synchronisation de l'horloge et estimation du temps de transit

Un client envoie un message TIMESTAMP REQUEST *à* un autre client pour demander à la machine de destination de lui transmettre sa valeur actuelle de l'heure du jour. La machine réceptrice répond en envoyant un TIMESTAMP REPLY *à* la machine émettrice.

Le format du message TIMESTAMP REQUEST et TIMESTAMP REPLY sont illustrés dans la figure ci-dessous:



Format du message ICMP de demande et de réponse de synchronisation de l'horloge.

Les champs SEQUENCE et IDENTIFIIER sont utilisés par la source pour associer les demandes aux réponses. Le champ TYPE identifie le message comme une demande (13) ou une réponse (14). Les autres champs spécifient les temps, en millisecondes à partir de minuit (GMT).

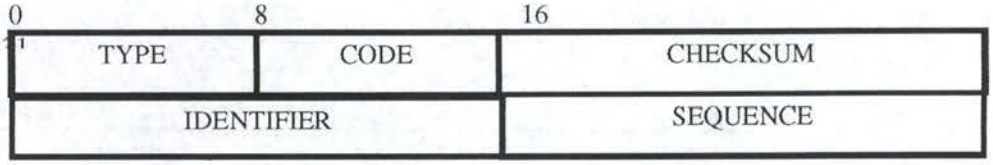
Le champ ORIGINATE est rempli par l'émetteur juste avant la transmission du paquet. Le champ RECEIVER est rempli immédiatement après la réception d'une demande, et le champ TRANSMIT est rempli immédiatement avant qu'une réponse soit transmise.

Les hôtes utilisent ces trois champs pour calculer les estimations des délais et pour synchroniser leurs horloges. Comme la réponse comprend le champ ORIGINATE TIMESTAMP, un hôte peut calculer le durée totale nécessaire pour la transmission d'une demande à une destination, sa transformation en une réponse et son retour.

La réponse transporte à la fois l'heure à laquelle la demande entre dans la machine réceptrice, ainsi que l'heure à laquelle la réponse sort, l'hôte peut calculer le temps de transit du réseau, et à partir de cela, estimer les différences des temps horloge des deux machines.

2.3.2.10 Obtention d'une adresse réseau

Les machines utilisent les messages ICMP de demande d'information pour obtenir une adresse Internet pour un réseau auquel ils sont reliés; c'est une alternative pour RARP. L'émetteur remplie une demande, l'envoie en mettant à zéro l'adresse réseau de destination IP, et attend une réponse. La réponse arrive avec la partie réseau de l'IP de l'émetteur remplit. Les passerelles répondent aux demandes des adresses de réseaux et envoient leurs réponses avec une bonne spécification des deux champs source et destination du datagramme IP. Le format des messages de demande et de réponse est illustré dans la figure ci-dessous:



Format des messages ICMP de demande et de réponse d'information

Les champs IDENTIFIER et SEQUENCE contiennent les valeurs que l'émetteur utilise pour associer des demandes à des réponses. Le champ TYPE spécifie si le paquet contient une demande (15) ou une réponse (16).

2.3.3 Protocoles de résolution d'adresses : *ARP* et *RARP*

2.3.3.1 Origine du problème

Considérons deux machines A et B sur un même réseau. Chacune d'entre elles possède une adresse Internet (respectivement IA et IB) et une adresse physique (respectivement PA et PB). L'objectif est de combiner les logiciels de bas niveau qui cachent les adresses physiques et de permettre aux programmes de haut niveau de travailler avec seulement des adresses Internet. La communication finale doit être, cependant, réalisé par des réseaux physiques utilisant n'importe quel type d'adresse physique fournit par le matériel. La question qui se pose est: si une machine A veut envoyer un paquet à une autre machine B sur le réseau qui les relie, et qu'elle ne dispose que de l'adresse Internet IB de B, comment peut elle transformer cette adresse en une adresse physique PB ?

Le problème de la transformation des adresses Internet en des adresses physiques est connu sous le nom du problème de résolution des adresses. Ce problème a été résolu de plusieurs façons. Quelques protocoles Internet gardent des tables dans chaque machine, ces tables contiennent des paires d'adresses Internet-physiques. D'autres ont résolu le problème en encodant les adresses physiques en des adresses Internet. En utilisant la méthode des tables nous pouvons rendre l'adressage Internet difficile.

En fait, à chaque hôte est assignée une adresse de 32 bits. Cette adresse est appelée l'adresse Internet. Ces adresses sont utilisées pour envoyer ou pour recevoir les paquets. Mais pour que deux machines puissent communiquer physiquement sur un réseau, ils ne peuvent utiliser que des adresses physique. Pour cela, l'adresse internet doit être convertie en une adresse physique. Le protocole ARP (pour *Address Resolution Protocol)* nous permet de résoudre ce problème.

2.3.3.2 Protocole ARP

2.3.3.2.1 Structure statique du protocole ARP

2.3.3.2.1.1 Encapsulation du message ARP

Quand les messages ARP se déplacent d'une machine à une autre, ils doivent être transportés dans des trames physiques. Ils sont ainsi encapsulés dans des trames Ethernet :

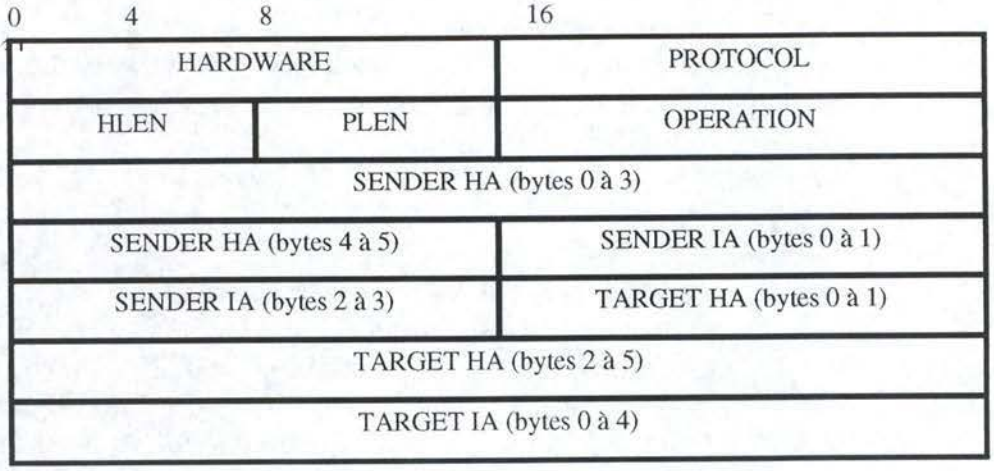
En-tête de trame	Message ARP complet traité comme une donnée

Pour voir si la trame transporte une requête ou une réponse ARP, l'émetteur affecte une valeur spéciale dans le champ TYPE de l'en-tête de la trame et place le message ARP dans le champ de donnée de la trame. Quand une trame arrive à un hôte, le système examine son type pour déterminer son contenu. Par exemple, sur Ethernet, les requêtes ARP ont un champ type d'une valeur 080616 (base 16) et les réponses ont un type d'une valeur de 803516. Ceux-ci sont des valeurs standards, affectées par l'autorité qui a établi les standards Ethernet.

2.3.3.2.1.2 Format du protocole ARP

Le format du message ARP est différent de celui des autres protocoles. Les données dans les paquets n'ont pas un en-tête de format fixe. Le message est conçu pour être utilisé par une variété de technologie de réseau, ainsi les premiers champs de l'en-tête contiennent des comptes qui spécifient les longueurs des champs suivants. En fait, ARP peut être utilisé avec des adresses physiques et des adresses de protocoles arbitraires.

L'exemple de la figure ci-dessous montre le format du message ARP d'une longueur de 28 octets utilisé sur le support Ethernet (où les adresses physiques sont de 48 bits de longueur), quand on résout les adresses du protocoles Internet du DARPA (d'une longueur de 4 octets). A la différence de la majorité des protocoles Internet, les champs de longueurs variables dans les paquets ARP ne s'alignent pas sur les limites de 32 bits, rendant ainsi le diagramme difficile à lire. Par exemple, l'adresse de la machine de l'émetteur SENDER HA, occupe 6 octets successifs, alors elle prend deux lignes dans le diagramme. Néanmoins, ce format a été choisi parce qu'il est un standard dans la littérature d'Internet.



Format des messages ARP/RARP utilisé pour la résolution d'adresse Internet => Ethernet.

Le champ HARDWARE spécifie le type de l'interface de hardware pour laquelle l'émetteur cherche une réponse; elle a une valeur de 1 pour le support Internet. Le champ OPERATION spécifie une requête ARP (1), ou une réponse ARP (2), ou une requête RARP (3), ou encore une réponse RARP (4). Les champs HLEN et PLEN permettent au protocole ARP d'être utilisé avec des réseaux arbitraires puisqu'ils spécifient la longueur de l'adresse physique de la machine et la longueur de l'adresse du protocole. L'émetteur fournit l'adresse de sa machine ainsi que son adresse Internet, s'ils sont connus, dans les champs SENDER HA et SENDER IA.

Quand il envoie une requête, l'émetteur fournit également l'adresse Internet de destination (ARP), ou l'adresse de la machine de destination (RARP) en utilisant les champs TARGET HA et TARGET IA. Une réponse transporte à la fois l'adresse physique et Internet de la machine de destination.

2.3.3.2.2 Résolution par liaison dynamique

Ethernet a des adresses physiques de 48 bits affectés par les vendeurs lors de la fabrication de l'interface. En remplaçant une interface, l'adresse physique de la machine change. En plus, comme l'adresse Ethernet est de 48 bits, on ne peut pas l'encoder en une adresse Internet de 32 bits.

Les concepteurs d'Internet ont trouvé une solution créative à ce problème de résolution d'adresse pour des réseaux de type Ethernet. La solution permet d'ajouter des nouvelles machines au réseau sans devoir recompiler le code, mais ne nécessite

pas la maintenance d'une base de données centralisée. Pour éviter la maintenance d'une table de correspondance, ils ont choisi d'utiliser un protocole de bas niveau pour relier les adresses dynamiquement. L'idée est simple: quand un hôte A veut résoudre l'adresse Internet IB , il diffuse un paquet spécial qui demande à l'hôte ayant cette adresse Internet de répondre avec son adresse physique, PB Tous les hôtes, y compris B, reçoivent la requête, mais seulement l'hôte B reconnaît son adresse Internet et envoie une réponse qui contient son adresse physique. Quand A reçoit la réponse, elle retient l'adresse physique de B et l'utilise pour envoyer le paquet Internet directement à B.

Ainsi, le protocole de résolution d'adresses (ARP) permet à un hôte de trouver l'adresse physique d'un autre hôte **sur le même réseau physique** en donnant seulement l'adresse Internet de ce dernier.

2.3.3.2.3 Sauvegarde des correspondances des adresses

Il peut paraître anormal que A diffuse un message pour dire à B "comment je peux vous atteindre" au lieu de diffuser directement le paquet qu'elle veut transmettre. Mais il y a une raison importante pour cet échange. La diffusion coûte cher si elle est utilisé chaque fois qu'une machine doit transmettre un paquet puisqu'elle nécessite le traitement d'un paquet de diffusion. Pour réduire les coûts de communication, les hôtes qui utilisent les requêtes ARP gardent les dernières correspondances d'adresses (Internet-physique) dans une cache, de façon à ce qu'ils n'utilisent pas ARP de façon répétitive. Quand l'hôte reçoit une réponse ARP, il enregistre le résultat dans cet endroit pour une utilisation ultérieure. Quand il veut envoyer un paquet à une destination quelconque, l'hôte va d'abord voir dans cet endroit si la correspondance existe avant d'envoyer une requête ARP. S'il trouve la correspondance voulue, il n'envoie pas le message de requête sur le réseau.

On peut raffiner ce procédé. Premièrement, si un hôte A doit utiliser un ARP pour envoyer un message à l'hôte B, il y a une grande probabilité que l'hôte B veuille répondre à A dans un futur proche. Si on anticipe les besoins de B, on peut éviter du trafic supplémentaire en incluant une correspondance d'adresses de A quand celle-ci envoie une requête à B. Deuxièmement, quand A diffuse sa requête initiale, toutes les machines sur le réseau la reçoivent et peuvent l'extraire et l'enregistrer chez eux. Troisièmement, quand une nouvelle machine vient s'ajouter au réseau, on peut éviter que tous les autres machines exécutent un ARP en diffusant la nouvelle paire (correspondance des adresses).

Ainsi, dans chaque message ARP diffusé on trouve l'adresse Internet de l'émetteur et son adresse physique correspondante; le récepteur met à jour les informations de correspondances d'adresses dont il dispose avant de traiter un paquet ARP.

2.3.3.2.4 Relation entre ARP et les autres protocoles

ARP fournit un mécanisme de transformation des adresses Internet en adresses physiques; ce mécanisme n'est disponible que pour certains réseaux. Il serait inutile si

tous les interfaces de réseaux comprenaient les adresses Internet. ARP impose un nouveau système d'adressage dépassant tout ceux utilisés par les machines.

Ainsi, ARP est un protocole de bas niveau qui cache l'adressage physique du réseau, pour nous permettre d'affecter des adresses Internet de notre choix à chaque machine. **Il est considéré comme une partie du système physique du réseau, et non comme une partie des protocoles Internet.**

2.3.3.2.5 Implémentation de ARP

ARP est divisé en **deux parties** :

- **Une partie** qui détermine les adresses physiques quand on envoie des paquets, et l'autre répond aux requêtes des autres machines. La résolution des adresses des paquets transmis semble assez simple, mais les petits détails compliquent l'implémentation.

L'hôte consulte la table de correspondance des adresses. S'il trouve l'adresse physique correspondante à l'adresse Internet en question, il l'extrait, place les données dans la trame en utilisant cette adresse et envoie la trame. S'il ne trouve pas la correspondance, il doit diffuser une requête ARP et doit attendre une réponse.

La diffusion d'une requête ARP pour trouver une correspondance d'adresses Internet peut devenir complexe. La machine de destination peut être hors d'usage ou encombrée. Dans ce cas, elle ne peut pas traiter la requête et donc l'émetteur ne reçoit pas une réponse ou en reçoit une mais en retard. La requête ARP diffusée peut être également perdue.

L'hôte doit ranger, entre temps, les paquets à émettre pour qu'ils soient transmis une fois l'adresse a été résolue. En fait, l'hôte doit décider s'il doit établir de requête ARP multiple ou pas, il doit savoir également s'il est permis d'envoyer plusieurs requêtes. L'hôte ne doit pas diffuser plusieurs requêtes ARP vers une adresse Internet de destination.

La valeur cachée peut être dépassée rendant ainsi le succès de la transmission impossible.

- La **deuxième partie** du code ARP s'occupe des paquets ARP qui arrivent du réseau. Sa cache est examinée pour voir si elle a déjà une entrée pour l'émetteur et si c'est le cas, cette entrée sera mise à jour en copiant l'adresse physique de l'émetteur du paquet. Le paquet sera traité par la suite.

L'hôte s'occupe de deux types de paquets ARP reçus. Il doit examiner les paquets reçus des requêtes ARP. Ces paquets doivent contenir la destination d'une requête d'une autre machine. Dans ce cas, une réponse est préparée et envoyée par le mécanisme ARP. Cette réponse contient son adresse physique. Sinon, le paquet doit contenir une

demande de correspondance avec une autre machine et peut donc être ignoré. Dans ce deuxième cas, quand une réponse ARP arrive et selon l'implémentation, une entrée de cache est créée puis remplie par la correspondance Internet-physique de l'émetteur avant le traitement. Si les réponses envoyées concement une requête, les paquets en attente seront transmis. Si l'hôte ne s'intéresse pas à la réponse il arrête simplement le traitement du paquet.

2.3.3.3 Protocole RARP

2.3.3.3.1 Introduction

Habituellement l'adresse Internet d'une machine se trouve rangée sur le disque dur. Le système d'exploitation, au démarrage, va charger ces informations qui se trouve sur le support auxiliaire. Le problème qui se pose est de savoir comment une machine sans disque peut déterminer sa propre adresse Internet ? Le problème est critique pour les stations de travail sans disque qui utilisent des adresses Internet pour communiquer avec le serveur de fichiers. Ces machines utilisent le protocole RARP (pour *Reverse Address Resolution Protocol) pour* résoudre ce problème et ainsi obtenir une adresse.

L'idée de trouver une adresse Internet est assez simple: la machine sans disque envoie une demande à un serveur et attends une réponse de ce dernier. On suppose que le serveur a un disque sur lequel il dispose d'une base de données des adresses Internet. Dans la demande, la machine qui veut connaître son adresse Internet doit s'identifier d'une manière unique de façon que le serveur puisse chercher l'adresse Internet correcte et envoyer une réponse. Les deux machines utilisent les adresses physiques du réseau durant leur bref communication. La machine sans disque ne connaît pas l'adresse physique du serveur, elle diffuse simplement une requête à toutes les machines sur le réseau local. Un ou plusieurs serveurs répondent à cette requête.

Quand une machine sans disque diffuse sa requête, elle doit s'identifier d'une façon unique. Une information que le système d'exploitation peut diffuser et qui peut identifier d'une manière unique la machine sur laquelle il s'exécute peut être n'importe quel information sur l'identification du matériel. Le numéro de série du CPU, par exemple, suffit. Mais une telle information peut être difficile à obtenir, et la longueur ou le format peut varier d'un CPU à un autre.

2.3.3.3.2 Reverse Address Resolution Protocol (RARP)

Une autre information qui peut identifier d'une manière unique une machine et qui est toujours disponible est son adresse physique de réseau. L'utilisation de cette information a deux avantages:

- cette adresse est toujours disponible puisqu'on peut l'obtenir de l'interface réseau du matériel sans être lié au système d'exploitation;
- comme l'information d'identification dépend du réseau et non pas du modèle ou du vendeur du CPU, toutes les machines sur un réseau donné vont fournir des identifiants uniques et uniformes.

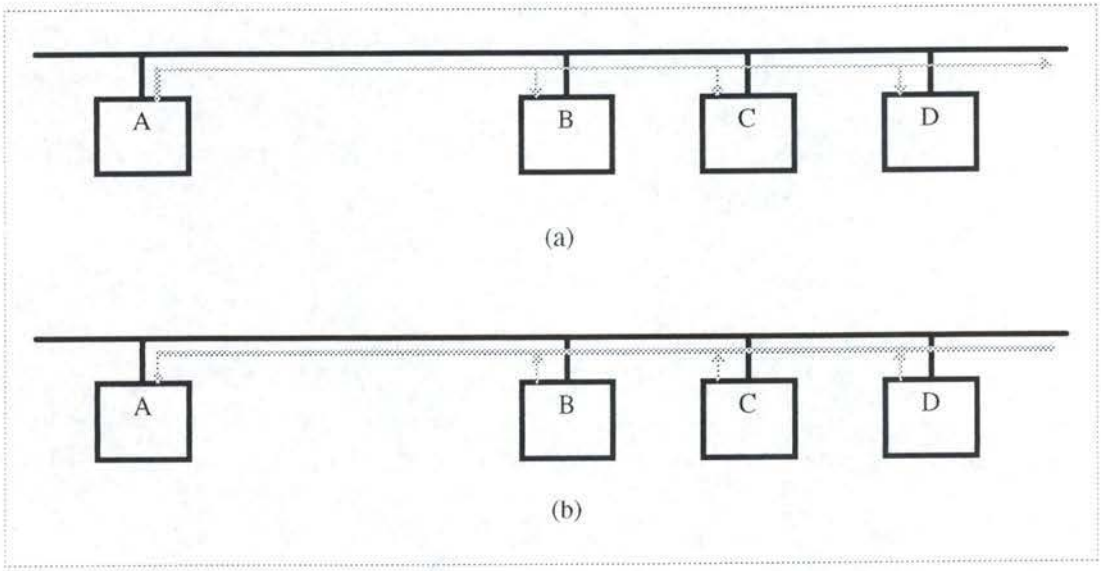
Ainsi, le problème devient une résolution d'adresses à l'envers: étant donné une adresse physique du réseau, un serveur la transfon-ne en une adresse Internet.

Le protocole que les machines sans disque utilise pour communiquer avec un serveur, qui peut leur fournir les adresses Internet, est appelé le *Reverse Address Resolution Protocol* (RARP). C'est une adaptation du protocole ARP utilise le même format de message. Le message RARP envoyé demande une adresse Internet est plus général que ce qu'on a décrit: il permet a une machine de demander assez facilement n'importe quelle autre adresse Internet et plusieurs types de réseaux physiques.

Comme ARP, le message RARP est envoyé d'une machine à une autre encapsulé dans la partie de données d'une trame Ethernet. Une trame Ethernet qui transporte la requête RARP a le préambule habituel, les adresses Ethernet source et destination, et le champ type du paquet au début de la trame. Le type de la trame contient une valeur qui identifie le contenu de la trame comme étant une requête RARP. La partie de donnée de la trame contient le message RARP de 28 octets.

On peut illustrer la façon dont un hôte utilise RARP: l'émetteur diffuse une requête RARP qui se spécifie comme étant la machine cible et fournit son adresse physique de réseau dans le champ matériel cible *(target hardware)*. Toutes les machines sur le réseau reçoivent la requête, mais seulement ceux qui sont autorisées de fournir le service RARP traitent la requête et envoient une réponse; de telles machines sont des serveurs RARP. Pour que le mécanisme RARP réussisse, le réseau devrait contenir au moins un serveur RARP.

La figure ci-dessous illustre l'utilisation de RARP:



En (a) l'hôte A diffuse une requête RARP en se spécifiant comme étant cible, et en (b) ces machines autorisées à fournir le service RARP répondent directement à A.

Les serveurs répondent aux requêtes en remplissant le champ adresse cible du protocole, en changeant le type de message (requête => réponse) et en envoyant la réponse directement à la machine qui a émis la requête. Cette machine reçoit des réponses de tous les serveurs RARP, même si la première suffit.

Toutes les communications entre l'hôte qui cherche son adresse Internet et le serveur qui le fournit doivent être effectuées seulement sur le réseau physique. En plus, le protocole permet à un hôte de demander une cible arbitraire. Ainsi, l'émetteur fournit son adresse du matériel séparé de l'adresse cible du matériel, et le serveur est prudent quand il envoie la réponse en utilisant l'adresse du matériel de l'émetteur.

Avec Ethernet, le champ pour l'adresse du matériel de l'émetteur peut paraître redondant car l'information est aussi contenue dans l'en-tête de la trame Ethernet. Cependant, le technologie Ethernet ne donne pas, généralement, un accès au système d'exploitation à l'en-tête physique de la trame.

2.3.3.3.3 Les problèmes de transactions avec RARP

Comme toutes les autres communications réseaux, les requêtes RARP peuvent être perdues ou corrompues. Le protocole RARP utilise seulement le réseau physique, son logiciel doit donc gérer les délais d'attente et les retransmissions. Généralement, RARP est seulement utilisé sur des réseaux locaux tel que Ethernet, où la probabilité d'un échec est faible. Cependant, si le réseau n'a qu'un seul serveur RARP, il n'est pas capable de manipuler le chargement, ainsi les paquets pourraient être perdus.

Plusieurs machines sans disque comptent sur RARP pour la mise en route et peuvent choisir de réessayer indéfiniment jusqu'à ce qu'ils reçoivent une réponse. D'autres implémentations annoncent un échec après seulement quelques essais pour éviter ainsi d'inonder le réseau avec un trafic de diffusion inutile. Sur Ethernet, la défaillance du réseau est moins probable qu'une surcharge du serveur. Si la retransmission RARP était plus rapide, on pourrait avoir des effets indésirables comme celui de l'inondation de serveurs, déjà encombrés, avec plus de trafic. Ainsi, un grand retard assure que les serveurs ont suffisamment de temps pour satisfaire la requête et renvoyer une réponse.

2.3.3.3.4 Les serveurs RARP

Le principal avantage d'avoir plusieurs machines fonctionnant comme des serveurs RARP est de rendre le système plus fiable. Si des serveurs sont en panne ou sont lourdement chargés, d'autres peuvent répondre à la requête. Ainsi, il est très probable que le service sera disponible. Cependant, l'inconvénient d'utiliser plusieurs serveurs est que quand une machine diffuse une requête RARP, le réseau devient surchargé puisqu'ils vont tous tenter de répondre. Sur un Ethernet, par exemple, en utilisant plusieurs serveurs RARP, la probabilité d'une collision augmente.

Il existe **deux possibilités** pour arranger le service RARP de façon à le rendre disponible et fiable sans contracter des coûts de plusieurs réponses simultanées. Ces deux possibilités ont pour but de retarder les réponses.

• Dans la **première solution**, un serveur principale est affecté aux machines qui font des requêtes RARP. Dans le cas normal, seul le serveur principal de la machine qui répond à sa requête RARP. Tous les autres serveurs reçoivent la requête et enregistrent seulement son heure d'arrivée. Si le serveur principal est non disponible, la machine émettrice va dépasser son délai d'attente de la réponse et va ensuite rediffuser la requête. Lorsque un autre serveur (non principal) reçoit une copie de la requête RARP dans le premier intervalle de temps, il répond.

• La **deuxième solution** utilise un plan similaire mais tente d'éviter la transmission simultanée de la réponse des autres serveurs. Chaque machine, non principale, qui reçoit une requête calcule un délai aléatoire et ensuite envoie une réponse. Normalement, le serveur principal répond immédiatement, et des réponses successives sont retardées de façon à ce que la probabilité qu'ils arrivent en même temps est faible. Quand le serveur principal est non disponible, la machine émettrice subit des petits retards avant de recevoir une réponse.

2.4 Protocoles de niveau Transport.

2.4.1 Protocole TCP

2.4.1.1 Introduction

2.4.1.1.1 Objectifs

Les systèmes de communication des ordinateurs jouent un rôle de plus en plus important dans les environnements militaires, gouvernementaux et civils. Comme ces réseaux de communication sont développés et déployés pour des buts stratégiques et tactiques, il est important de fournir des moyens de les interconnecter et de fournir des protocoles de communication standards qui peuvent supporter une large gamme d'applications.

Le protocole TCP *(Transmission Control Protocol),* est considéré comme étant un protocole de communication très fiable entre ordinateurs hôtes dans des réseaux utilisant la commutation par paquets, et dans l'interconnexion de ces réseaux. Il est orienté connexion et adapté aux systèmes de couche de protocoles hiérarchiques. TCP fournit une communication inter-processus fiable entre des paires de processus d'ordinateurs hôtes reliés à d'autres réseaux de communication. En principe, TCP devrait être capable d'opérer au-dessus d'un large gamme de systèmes de communication tels que les réseaux à commutation par paquets, à commutation de circuits etc.

Ainsi, le protocole de contrôle de transmission, TCP, définit un service clé offert par Internet, à savoir, une livraison fiable (*reliable*) des données. Ce service est le plus important et le plus connu des services Internet. TCP offre une connexion en *Full Duplex* entre deux machines leur permettant d'échanger de grands volumes de données. Comme il utilise un système de fenêtrage, TCP peut également rendre l'utilisation du réseau efficace.

Le service de TCP est flexible, il peut s'adapter à une grande variété de systèmes de transmission supportant IP. Il offre également un système de contrôle de flux permettant la communication entre des systèmes ayant des vitesses variées.

L'unité de base de transfert utilisé par TCP est le segment. Ces segments sont également utilisés pour faire passer les informations de contrôle, par exemple, en permettant au logiciel TCP sur les deux machines d'établir ou d'interrompre une connexion. Le format du segment permet également à une machine d'ajouter *(piggybacking)* dans l'en-tête du segment de donnée circulant dans un sens, un accusé de réception d'une donnée reçue (circulant dans le sens opposé).

TCP implémente le contrôle de flux en informant le récepteur de la quantité de données qu'il pourrait transmettre. Il supporte également des messages hors bande en utilisant un mécanisme de transmission de données urgentes et permet une livraison forcé en utilisant le mécanisme *push*. Tous ces mécanismes vont être développés dans les paragraphes qui suivent.

Le protocole TCP est un protocole indépendant malgré qu'il soit présenté comme une partie du protocole Internet, il est également un protocole général qui peut être utilisé dans d'autres systèmes. Par exemple, il est possible de l'utiliser dans un réseau du type Ethernet, supportant IP, ou dans des réseaux Internet plus complexes.

TCP offre des spécifications des formats de données et des accusés de réception que peuvent échanger deux ordinateurs pour accomplir un transfert fiable. Il constitue l'ensemble des procédures utilisées par les ordinateurs pour assurer l'arrivé correcte des données. Il spécifie également la manière dont le logiciel TCP distingue un programme d'application cible parmi plusieurs sur une machine donnée, ainsi que la manière par laquelle des machines en communication retrouvent des erreurs telles que les pertes ou les duplications de paquets.

2.4.1.1.2 Problèmes de livraison

Les paquets peuvent être perdus ou détruits quand il y a des interférences, quand le matériel tombe en panne ou quand les réseaux deviennent trop chargés. Les réseaux qui routent les paquets peuvent les livrer dans un désordre, ou avec un retard considérable ou peuvent également générer des duplications. De plus, les technologies des réseaux peuvent exiger des tailles optimales de paquets et peuvent poser d'autre contraintes nécessaires pour un transfert efficace.

Un des buts des recherche des protocoles de réseaux était de trouver une solution générale au problème qui consiste à fournir une livraison fiable des données. La construction d'un seul programme de protocole pour que tous les programmes d'application puissent l'utiliser a été envisagée. Un seul protocole peut ainsi contribuer à isoler les programmes d'application des détails du réseau et, par conséquent, rend la définition d'une interface uniforme possible pour le service de transfert de données.

2.4.1.1.3 Liaison entre interface d'application et service fiable de livraison

L'interface entre les programmes d'application et le service Internet de livraison de données possède **cinq** caractéristiques:

1. **Orientation des données:** Quand deux programmes d'application transfèrent de larges volumes de données, on considère ces données comme des flots de bits, divisés en octets de 8-bits ou bytes. Le service de livraison de données sur la machine de destination passe au récepteur exactement la même séquence d'octets que celle que l'émetteur sur la machine source a transmis.

2. **Les connexions:** Le transfert d'un flot de données est analogue à un appel téléphonique. Avant le début du transfert de données, l'émetteur et le récepteur interagissent avec leurs systèmes d'exploitation respectifs. Ceux-ci les informent des demandes de transfert de données.

   Généralement, la machine émettrice fait un appel à la machine réceptrice, qui accepte cet appel. Les modules du logiciel du protocole dans les deux systèmes d'exploitations communiquent entre eux en envoyant des messages sur l'Internet vérifiant que le transfert est autorisé, et que les deux machines sont prêtes.

   Une fois que ces détails sont réglés, les modules des protocoles informent les programmes d'application qu'une connexion a été établie et que le transfert peut commencer. Pendant le transfert, le logiciel du protocole sur les deux machines vérifie que les données sont reçues correctement tout en continuant la communication. Si la communication échoue, les deux machines détectent la panne et la signale au programme d'application approprié.

   On utilise parfois le terme circuit virtuel pour décrire de telle connexion car malgré le fait que des programmes d'application voient la connexion comme un circuit matériel, la fiabilité est une illusion fourni par le service de livraison de données.

3. **Transfert avec mémoire-tampon:** Les programmes d'application envoient un flot de données dans le circuit virtuel. Ce transfert se fait byte par byte de façon répétitive. Pour transmettre les données, chaque application utilise les tailles qui lui conviennent. Ces tailles peuvent même être d'un byte. A la réception, le système livre les bytes du flot de données dans l'ordre exacte de leur transmission, et une fois qu'ils sont vérifiés ils seront disponibles au programme d'application récepteur.

   Le logiciel du protocole peut diviser le flot de données en paquets indépendants différents de ceux qui sont transmis par le programme d'application source. Pour rendre le transfert plus efficace et pour minimiser le trafic du réseau, un certain nombre de données d'un flot sont réassemblés pour remplir un datagramme d'une taille raisonnable

avant la transmission. Ainsi, même si le programme d'application génère des flots d'un byte, le transfert dans l'Internet peut être très efficace.

Pour des applications où les données doivent être livrées même si elles ne remplissent pas la mémoire-tampon, le service de transmission de flots de données offre un mécanisme *appelé push* qui est utilisé par les applications pour forcer le transfert. Du côté de l'émetteur, un *push* force le logiciel du protocole à transférer toutes les données qui ont été générées sans attendre le remplissage de la mémoire-tampon. Lorsqu'elles atteignent leur destination, le *push* va obliger le TCP à mettre les données immédiatement à la disposition de l'application. Un *push* garantit le transfert de toutes les données. Ainsi, même si la livraison est forcée, le logiciel du protocole peut choisir de diviser le flot de données à sa façon.

4. **Flots de données non structurés:** Le service Internet de livraison de données ne respecte pas les flots de données structurées. Les programmes d'applications utilisant un service de livraison de données doivent bien comprendre le contenu du flot de données et doivent également convenir d'un format avant de commencer une connexion.

5. **Connexion en Full Duplex:** Les connexions offertes par le service Internet de livraison de données permettent un transfert simultané dans les deux sens. Une telle connexion est appelé *Full Duplex*. Du point de vue d'un processus d'application, une connexion en Full Duplex consiste en deux flots de données indépendants circulant dans les deux sens sans aucune interaction apparente. Ce service de livraison de données permet à un processus d'application de mettre fin à la circulation d'un flot de données dans un sens, bien que les données continuent à circuler dans l'autre. L'avantage d'une connexion Full *Duplex* est que le système peut envoyer des informations de contrôle à la source dans des datagrammes transportant des données. Un tel mécanisme s'appelle le *piggybacking, son* avantage est qu'il réduit le trafic dans le réseau.

2.4.1.2 Segments, séquences, et numéros de séquences

La séquence d'octets qui doit être transmise est divisée en segments. Chaque segment est envoyé dans un datagramme IP. TCP utilise un mécanisme de fenêtrage pour résoudre deux problèmes importants: l'efficacité de la transmission et le contrôle de flux. Ce mécanisme envoie plusieurs paquets avant de recevoir un accusé de réception. En faisant ainsi, le débit totale augmente puisque le réseau reste occupé.

Le protocole de la technique de fenêtrage résout également le problème de contrôle de flux de bout en bout, en permettant au récepteur de restreindre la transmission jusqu'à ce qu'il dispose d'une mémoire-tampon suffisante pour recevoir plus de données.

Le mécanisme de fenêtrage est utilisé au niveau byte, et pas au niveau paquet. Les bytes des données sont numérotés séquentiellement, l'émetteur doit garder trois pointeurs associés à une connexion et qui définissent la fenêtre. Le premier pointeur marque la partie gauche de la fenêtre, séparant les bytes qui ont été transmis et qui ont reçus leurs accusés de réception, des bytes qui vont être transmis. Un second pointeur marque la partie droite de la fenêtre et définit le byte le plus élevé dans la séquence. Ce byte peut être transmis avant que d'autres accusés de réception soient reçus. Le troisième pointeur marque les limites qui séparent les bytes de la fenêtre qui ont déjà été transmis de ceux qui ne le sont pas encore. Un exemple de fenêtrage est illustré dans les figures ci-dessous.

Le caractère *Full Duplex* de la transmission signifie que deux transferts peuvent s'effectuer simultanément sur chaque connexion, et dans les deux sens. Les transferts sont totalement indépendants car à tout moment les données peuvent circuler dans un ou dans les deux sens. Ainsi, le logiciel de TCP maintient deux fenêtres pour chaque connexion, une est utilisée pour les données à transmettre et l'autre pour les données reçues.

2.4.1.3 Variation de la taille de la fenêtre et le contrôle de flux

TCP permet la variation de la taille des fenêtres au cours du temps. Chaque accusé de réception contient une annonce de fenêtre *(window advertisement)* qui indique le nombre additionnel de bytes de données que le récepteur est prêt à accepter. En réponse à une annonce d'accroissement de la fenêtre, l'émetteur augmente la taille de sa fenêtre et procède à l'émission des octets dont il n'a pas encore reçu les accusés de réception. Si, par contre, il reçoit une information indiquant un rétrécissement de la fenêtre, l'émetteur réduit la taille de sa fenêtre et ne transmet plus les bytes qui dépassent les limites annoncées. Les annonces accompagnent les accusés de réception, ainsi, la taille de la fenêtre change quand celle-ci se déplace à droite.

L'avantage d'utiliser une taille variable de la fenêtre est de pouvoir contrôler le flux de données ainsi que la fiabilité du transfert. Si l'espace de la mémoire-tampon du récepteur commence à se remplir, il ne peut plus tolérer davantage de paquets, alors il envoie une annonce de petite fenêtre. Dans le cas extrême, le récepteur annonce une taille de fenêtre nulle pour stopper toutes les transmissions. Et quand l'espace de sa mémoire-tampon se vide, le récepteur annonce une taille de fenêtre non nulle pour déclencher de nouveau le flux de données.

Il est important d'avoir un mécanisme de contrôle de flux dans un environnement Internet. Car dans un tel environnement des machines de différentes vitesses et de différentes tailles communiquent sur des réseaux et à travers des passerelles qui ont elles même des vitesses et des capacités différentes. Il existe deux problèmes indépendants liés au contrôle de flux:

1. Les protocoles Internet nécessitent un contrôle de flux de bout en bout entre la source et la destination. Quand un mini-ordinateur, par exemple, communique avec un *mainframe, le* mini-ordinateur doit régler l'afflux de données, pour éviter que le système soit rapidement envahi. Ainsi, TCP

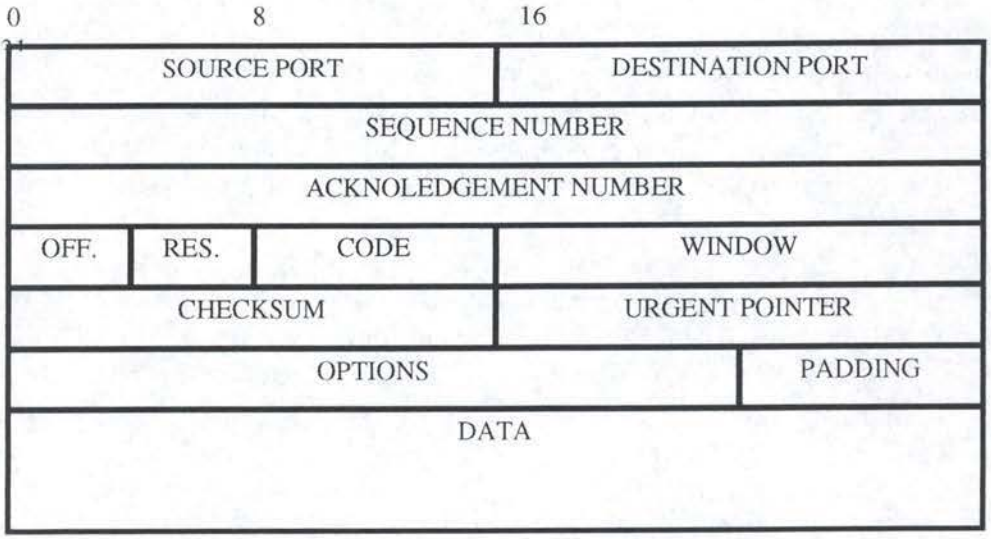
doit implémenter un contrôle de flux de bout en bout pour garantir une transmission fiable.

2. Les protocoles Internet nécessitent un mécanisme de contrôle de flux qui permet aux machines intermédiaires, comme les passerelles, de contrôler une source qui transmet plus de trafic qu'elles ne peuvent tolérer.

TCP combine la technique de fenêtrage avec le mécanisme de contrôle de flux pour résoudre le problème de contrôle de flux de bout en bout. Le deuxième problème est résolu par ces machines intermédiaires en utilisant les messages *source quench* d'ICMP.

2.4.1.4 Format du segment TCP

L'unité de transfert entre les modules TCP de deux machines est appelée segment . Des segments sont échangés pour établir une connexion, transférer des données, transmettre des accusés de réception, annoncer la taille de la fenêtre et pour fermer une connexion. TCP utilise la technique du *piggybacking*. Cette technique permet la transmission d'un accusé de réception d'une machine A vers une machine B dans le même segment que celui des données. Le format d'un segment TCP est illustré dans figure ci-dessous:



Format du segment TCP.

Chaque segment est divisé en deux parties, l'en-tête suivie de la zone de données. L'en-tête est souvent appelé le *TCP header*. Les champs SOURCE PORT et DESTINATION PORT de l'en-tête contiennent les numéros d'accès qui identifient les **programmes d'application** sur les machines connectées.

Les segments TCP sont transmis comme les datagrammes Internet. L'en-tête transporte plusieurs champs d'information. Un en-tête TCP est encapsulé dans la zone de données du paquet IP, il fournit des informations spécifiques au protocole TCP.

Le champ SEQUENCE NUMBER identifie la position de la donnée du segment dans la séquence de bytes transmise par l'émetteur. Le champ ACKNOWLEDGEMENT NUMBER identifie la position du byte le plus élevé que la source a reçu. On remarque que le numéro de séquence fait référence aux flots de données circulant dans le même sens que le segment, par contre le numéro de l'accusé de réception fait référence aux flots de données circulant dans le sens opposé du segment.

Le champ OFF. pour *offset* contient un entier de 4 bits, il indique l'emplacement des donnée dans le segment. Cet entier est nécessaire car le champ OPTIONS varie en longueur, il dépend de l'option choisie. Le champ RES. est réservé pour une utilisation future.

Certains segments transportent seulement un accusé de réception, d'autres transportent des données. Il y a des segments qui transportent des demandes d'ouverture ou de fermeture de connexion.

Le logiciel de TCP utilise un champ CODE de 6 bits pour déterminer le but et le contenu du segment. Les 6 bits indiquent comment interpréter d'autres champs de l'en-tête. Le tableau suivant résume les différents codes de contrôle:

Bit (de gauche à droite)	Explication
URG	Champ valide de pointeur urgent
ACK	Champ valide d'accusé de réception
PSH	Ce segment demande un *push*
RST	Remise à zéro de la connexion
SYN	Numéros de séquences synchronisés
FIN	L'émetteur a atteint la fin de l'émission

Chaque fois que le logiciel de TCP transmet un segment, il annonce le nombre de données qu'il est prêt à accepter en spécifiant la taille de sa mémoire-tampon dans le champ WINDOW. L'annonce de la taille de la fenêtre est un autre exemple de l'utilisation de la technique du *piggybacking* puisqu'elle accompagne tous les segments y compris ceux transportant des données ou seulement des accusés de réception.

TCP permet à l'émetteur de spécifier le caractère urgent de quelques données, c'est-à-dire que les données doivent être transmises dans les plus brefs délais.

Les données urgentes contiennent des messages différents des données normales. Par exemple, un trafic urgent pourrait inclure des signaux d'interruption tapés au clavier. Un tel trafic est souvent appelé *out of band traffic*.

Le détail exact de la façon dont TCP informe le programme d'application de l'urgence de la donnée dépend du système d'exploitation de l'ordinateur. Le mécanisme utilisé pour marquer les données urgentes utilise les deux champs: URG et URGENT POINTER. Quand le bit URG est mis à 1, le pointeur urgent spécifie une position dans la séquence de bytes où se terminent les données urgentes.

Le logiciel de TCP utilise le champ OPTIONS pour communiquer avec la machine réceptrice. En particulier, un récepteur peut spécifier la taille maximale du segment qu'il désire recevoir. Il est donc important de permettre au récepteur de spécifier une taille maximale d'un segment surtout si un petit ordinateur reçoit des données d'un plus grand ordinateur.

Il est difficile de choisir la bonne taille maximale du segment car les performances peuvent être faibles pour des tailles de segments très larges ou très petites. Par conséquence, quand la taille du segment diminue, l'utilisation du réseau diminue également. Ceci est dû au fait que les segments TCP qui circulent dans le réseau sont encapsulés dans des datagrammes IP, qui sont eux même encapsulés dans des trames physiques du réseau.

Ainsi, chaque segment a au moins 40 octets d'en-tête (TCP et IP) en plus de la donnée, et les datagrammes transportant un octet de données utilisent au plus 1/40 de la largeur de bande du réseau pour les données utilisateurs. D'autre part, des tailles de segments extrêmement larges produisent également des faibles performances. Les segments larges aboutissent à de larges datagrammes qui sont fragmentés avant la transmission. Les fragments, contrairement aux datagrammes, ne sont pas indépendants; tous les fragments doivent arriver ou doivent tous être retransmis. Si la probabilité de la perte d'un paquet est non nulle, l'augmentation de la taille du segment au dessus du seuil de fragmentation diminue le débit.

2.4.1.5 Riability

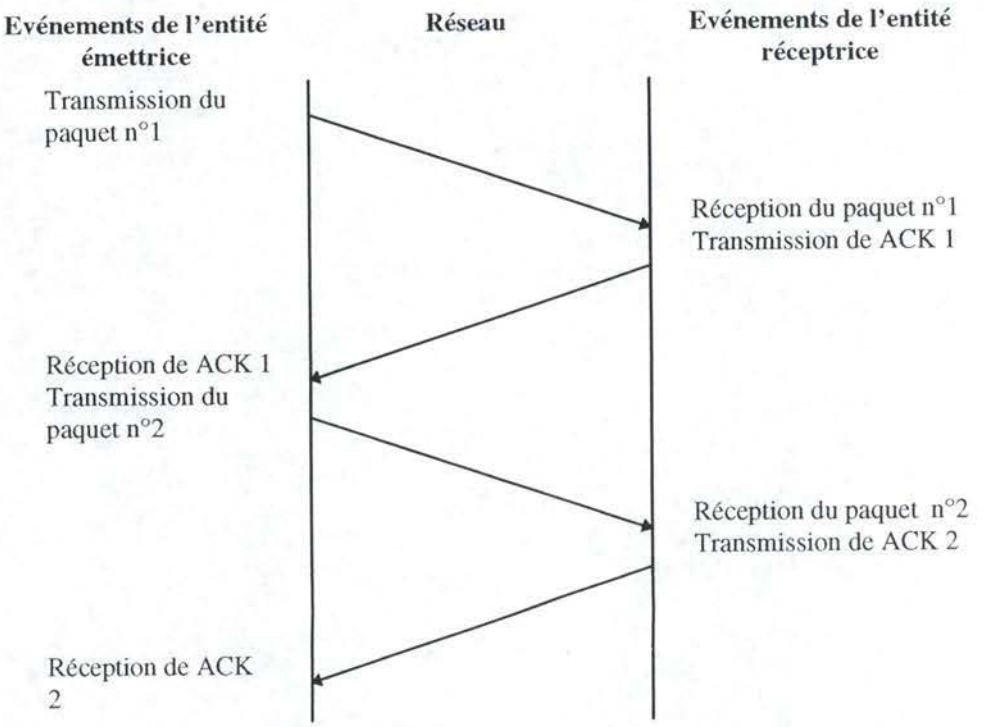
Le service TCP garantit une livraison sans duplication et sans perte de données entre deux machines. Pour ce faire, TCP utilise une seule technique fondamentale connu sous le *nom positive acknowledgement with retransmission.*

Cette technique exige du récepteur de communiquer avec la source, en lui envoyant un accusé de réception chaque fois qu'il reçoit des données. L'émetteur garde une trace de chaque paquet qu'il transmet et attend un accusé de réception avant de transmettre le prochain paquet. Il déclenche également une horloge quand il envoie un paquet et retransmet ce même paquet si le délai expire avant l'arrivé de son accusé de réception (évitant ainsi l'effet ping-pong).

Cet échange est illustré dans la figure (a) ci-dessous. les événements dans l'entité TCP émettrice et réceptrice se trouvent à gauche et à droite. Les lignes

diagonales qui traversent le milieu indiquent le transfert d'un message à travers le réseau.

La figure (b) utilise le même format de diagramme et montre ce qui se passe quand un paquet est perdu ou corrompu. L'émetteur déclenche une horloge après la transmission d'un paquet, quand le délai expire, il suppose que le paquet est perdu et le retransmet.



Simple transfert de données avec accusé de réception.

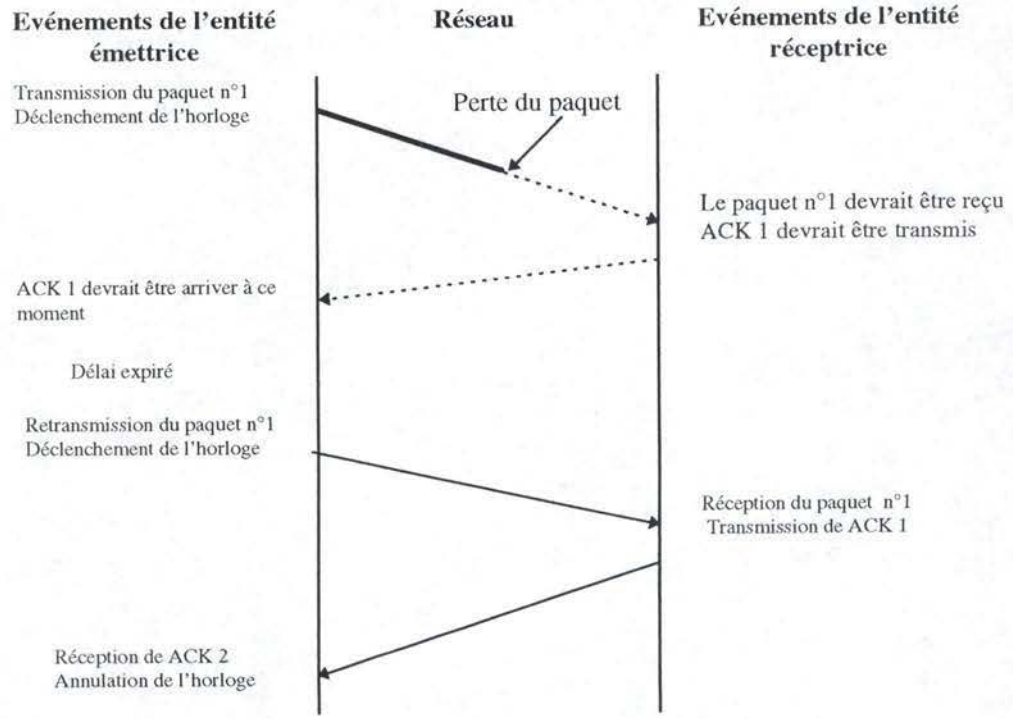
Le protocole utilise la technique *positive acknowledgement with retransmission* avec laquelle l'émetteur attend un accusé de réception (ACK) pour chaque paquet transmis; mais ceci n'implique pas qu'il faut attendre l'ACK avant d'envoyer le paquet suivant.

Les lignes verticales représentent le déroulement chronologique des événements dans chaque site.

Un autre problème de fiabilité apparaît quand un système de livraison duplique les paquets. Cette duplication peut également se présenter quand il existe de grands retards sur les réseaux, ceci implique des retransmissions prématurées. Il faut être très prudent pour résoudre ce problème car on peut avoir une duplication aussi bien des paquets que des accusés de réception.

Les protocoles de fiabilité détectent habituellement la duplication des paquets en assignant à chaque paquet un numéro de séquence et en exigeant du récepteur de se rappeler de ce nombre qu'il a reçu.

Pour éviter la confusion causé par les accusés de réception des paquets retardés ou dupliqués, *les positive acknowledgement protocol* envoient des nombres de séquence en retour dans l'accusé de réception, ainsi le récepteur peut correctement associer ces accusés aux paquets transmis.



Transfert avec perte de données.

Le dépassement du délai et la retransmission qui se produit quand un paquet est perdu, sont schématisé dans la figure ci dessus. Les lignes en pointillé indiquent le temps nécessaire pour la transmission d'un paquet et de son accusé de réception, si le paquet n'était pas perdu.

2.4.1.6 Technique de fenêtrage

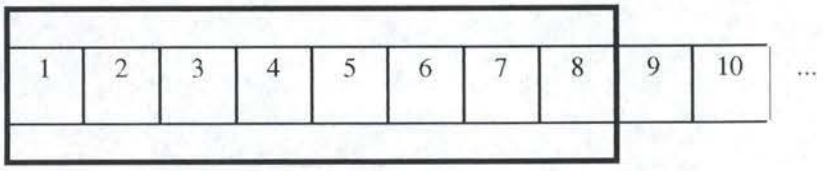
Pour rendre la transmission des flots de données efficace, on utilise la technique de fenêtre à glissière ou de fenêtrage. L'émetteur, après avoir transmis un certain nombre de paquets, attend un accusé de réception avant de transmettre un autre.

Comme la figure (a) ci-dessus le montre, à tout moment, les données circulent entre les machines dans un seul sens, même si le réseau a une capacité de communication simultanée et bidirectionnelle *(Full Duplex)*. Le réseau est souvent en repos si les machines tardent pour répondre. Par exemple, pendant que les machines calculent les routes ou les sommes de contrôle.

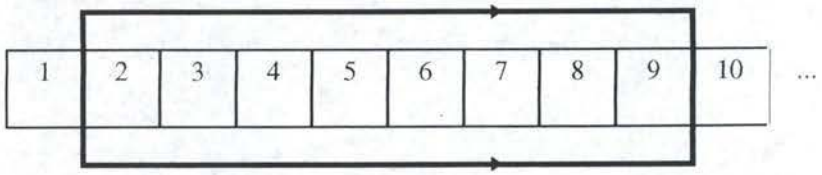
Si on dispose d'un réseau ayant des retards de transmissions assez longs, le problème devient plus claire: un *simple positive acknowledgement protocol* gaspille une grande quantité de la capacité du réseau car il doit retarder l'émission d'un

nouveau paquet jusqu'à ce qu'il reçoive l'accusé de réception du dernier paquet transmis.

La technique de fenêtrage est une forme plus complexe que la technique du *positive acknowledgement and retransmission.* Les protocoles de fenêtrage utilisent mieux la largeur de bande du réseau car ils permettent à l'émetteur de transmettre plusieurs paquets sans attendre un accusé de réception. Supposons qu'on dispose d'une séquence de paquets qu'on va transmettre. Le protocole va placer une petite fenêtre sur la séquence et transmet tous les paquets qui se trouvent à l'intérieur de cette fenêtre. Dans la figure (a) ci-dessous on a un protocole de fenêtrage à huit paquets et dans la figure (b) la fenêtre glisse de telle façon que le paquet 9 puisse être transmis et ceci après avoir reçu un accusé de réception du paquet 1.



(a) : fenêtre initiale



(b) : glissement de la fenêtre

Les fenêtres de l'exemple ci-dessus ont une taille de 8 paquets, l'émetteur peut transmettre 8 paquets avant de recevoir un accusé de réception. Une fois que l'émetteur reçoit l'accusé de réception du premier paquet, il fait glisser la fenêtre d'un paquet vers la droite, comme le montre la figure ci-dessus, et transmet le paquet suivant. La fenêtre continue à glisser tant que des accusés de réception sont reçus. La performance de ce protocole dépend de la taille et la vitesse à laquelle le réseau reçoit les paquets.

Le concept clé est que l'émetteur peut transmettre tous les paquets qui se trouvent à l'intérieur de la fenêtre sans attendre un accusé de réception. Ainsi, en augmentant la taille de la fenêtre, il est possible d'éliminer complètement le temps perdu par le réseau. Et donc avec un protocole de fenêtrage optimale, le réseau sera complètement saturé avec des paquets, et on obtient un plus grand débit qu'avec un simple protocole de fenêtrage de taille 1.

Généralement, ce protocole doit toujours se rappeler des paquets qui ont reçus leurs accusés de réception et tient pour chaque paquets, n'ayant pas encore reçu un

accusé, un *timer*. Si le paquet est perdu, le délai sera expiré et l'émetteur retransmet ce paquet. Quand l'émetteur fait glisser la fenêtre, il dépasse tous les paquets dont l'accusé de réception a déjà été reçu.

Du côté du récepteur, le système garde une fenêtre analogue, pour recevoir et accuser réception des paquets qui arrivent. Ainsi, la fenêtre partitionne la séquence de paquets en trois groupes:

- celui qui se trouve à gauche de la fenêtre contient les paquets qui sont déjà transmis avec succès, reçus et on a eu leurs accusés de réception;
- ceux qui sont à droite ne sont pas encore transmis
- et ceux qui se trouvent à l'intérieur de la fenêtre sont en cours de transmission.

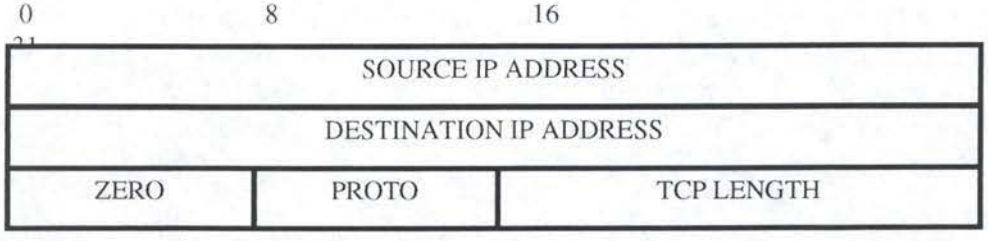
Le paquet ayant le numéro le plus bas dans la fenêtre est le premier paquet déjà transmis de la séquence dont l'accusé de réception n'est pas encore reçu.

2.4.1.7 Calcul de la somme de contrôle

Le champ CHECKSUM dans l'en-tête du segment TCP contient un entier de 16 bits utilisé pour vérifier l'intégrité de l'en-tête du segment et de la donnée. Pour calculer cette somme de contrôle, TCP sur la machine émettrice prépare un pseudo en-tête du segment, ajoute plusieurs bytes de valeur nulle pour remplir le segment complet *(header + data)* afin d'arriver à un multiple de 16 bits, et calcule la somme de contrôle de tout le résultat.

TCP ne compte pas les zéros ajoutés dans la longueur et ne les transmet pas. Il suppose également que le champ CHECKSUM vaut zéro pour faciliter le calcul de la valeur de contrôle. Comme pour d'autres valeurs de contrôle, TCP utilise une arithmétique à 16 bits. Au site récepteur, le logiciel de TCP exécute le même calcul pour vérifier si le segment reçu est intacte.

Le but de l'utilisation d'un pseudo en-tête est exactement le même que dans UDP (User Datagram Protocol). Il permet au récepteur de vérifier que le segment a atteint correctement sa destination, celle-ci se compose de l'adresse Internet de l'hôte et du numéro d'accès du protocole. La figure ci-dessous illustre le format du pseudo en-tête utilisé dans le calcul de la somme de contrôle:



Format du pseudo header.

Dans le *pseudo header*, le champ PROTO spécifie le type du protocole utilisé, et le champ TCP LENGTH spécifie la longueur totale du segment TCP.

Le récepteur extrait l'information du datagramme IP qui transporte le segment, analyse tous ses champs, recalcule la somme de contrôle pour la vérification et enfin enlève les zéros ajoutés.

2.4.1.8 Accusé de réception et retransmission

Comme TCP transmet les données dans des segments de longueurs variables, les accusés de réception s'appliquent à une position dans la séquence, et pas aux paquets ou aux segments. Chaque accusé de réception spécifie un byte plus grand que la plus haute position de byte qui a été reçue.

L'émetteur reçoit des *feed-back* continus du récepteur chaque fois qu'il progresse dans la séquence. Ainsi, les accusés de réception spécifient toujours le numéro du prochain byte que le récepteur s'attend à recevoir. Le mécanisme des accusés de réception TCP est appelé Il *cumulative* car il donne le nombre de séquences qui ont été accumulées. Ces accusés de réception cumulatifs ont à la fois des avantages et des inconvénients.

Un des avantages est que les accusés de réception sont à la fois facile à générer et non ambigus. Un autre avantage est que les accusés de réception perdus ne forcent pas nécessairement la retransmission. L'émetteur ne reçoit pas des informations concernant toutes les transmissions réussies, mais seulement une seule position dans la séquence qui a été reçue.

2.4.1.9 Délai d'attente et Retransmission

TCP s'attend à recevoir des accusés de réception de la destination chaque fois que ce dernier reçoit des données. Chaque fois qu'il envoie un segment, TCP déclenche une horloge et attend un accusé de réception. Si le délai est expiré avant l'arrivée de l'accusé de réception du segment de données, TCP suppose que le segment est perdu ou corrompu et le retransmet.

L'algorithme de retransmission TCP diffère des autres algorithmes utilisés dans plusieurs protocoles de réseaux. TCP est destiné à des environnements Internet où le chemin entre deux machines peut traverser un seul réseau à grande vitesse, où plusieurs passerelles à travers plusieurs réseaux intermédiaires. Il est donc impossible de connaître à priori à quelle vitesse les accusés de réception arrivent à la source. Le retard pris dépend du trafic, et le temps de transmission d'un segment et la réception de son accusé de réception sont variables.

TCP reçoit les différents délais Internet, et utilise un algorithme adaptatif avec ces délais. Pour collecter les données nécessaires pour l'algorithme adaptatif, TCP enregistre le moment auquel chaque segment est transmis, et le moment d'arrivé de

l'accusé de réception des données de ce segment. Avec ces deux moments, TCP calcule le temps écoulé connu sous le nom de *round trip time*. Chaque fois qu'il dispose d'un nouveau *round trip time*, TCP ajuste sa moyenne de durée de transfert.

Généralement, le logiciel de TCP garde une durée de transfert moyenne comme une référence et utilise les nouvelles durées pour modifier lentement cette référence. En bref, TCP collecte les différents intervalles de temps rencontrés dans l'environnement Internet, il utilise un algorithme de retransmission adaptatif qui contrôle ces délais et ajuste son paramètre du délai d'attente.

2.4.1.10 Réponse à l'encombrement

TCP considéré l'interaction entre les deux extrémités d'un connexion et les délais de communication entre eux. En pratique, cependant, TCP doit également réagir à l'encombrement de l'Internet. L'encombrement a pour conséquence des retards sévères causés par une surcharge de datagrammes dans un ou plusieurs points de commutation (dans les passerelles par exemple).

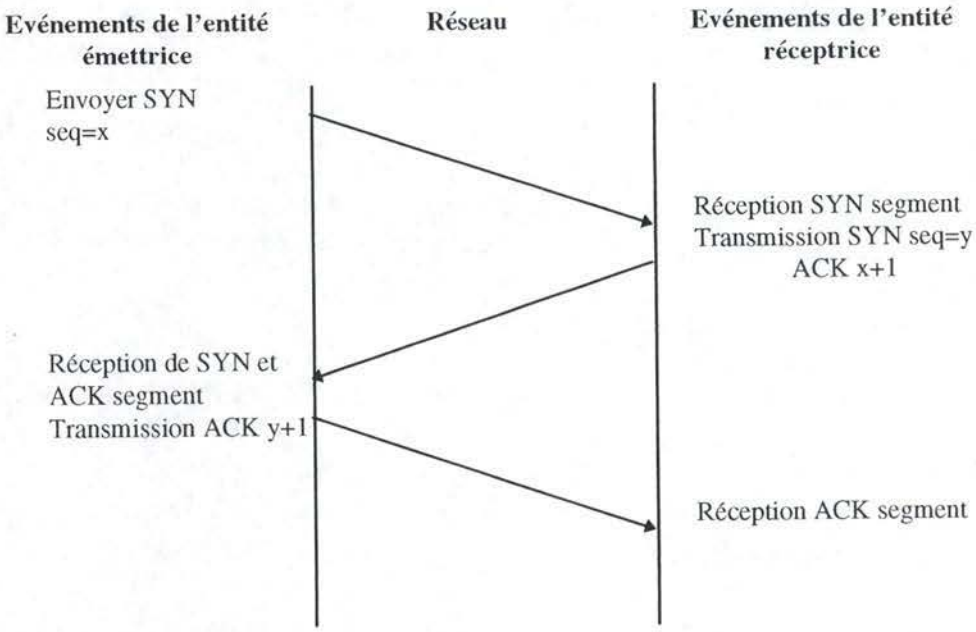
Quand il y a encombrement, les retards augmentent et la file d'attente des datagrammes devant les passerelles s'agrandit. Chaque passerelle possède une capacité de chargement limitée, il y a donc une concurrence pour les datagrammes. Dans le pire des cas, le nombre total de datagrammes qui arrivent à une passerelle encombrée augmente jusqu'à ce que la passerelle atteigne sa capacité maximale et commence à écarter des datagrammes.

Quand il y a un encombrement, TCP répond en réduisant le taux de transmission. Les passerelles peuvent utiliser des techniques telle que ICMP *(Internet Control Message Protocol)* pour informer les hôtes qu'un encombrement s'est produit. Mais les protocoles de niveau transport peuvent détecter l'encombrement automatiquement en observant l'augmentation des durées des délais de transfert.

Si la conception de TCP était pauvre, l'encombrement serait aggravé puisque l'augmentation des délais et les fréquentes retransmissions de segments seraient ignorés. Cependant, un TCP bien conçu aide à alléger l'encombrement.

2.4.1.11 Etablissement d'une connexion

Pour établir une connexion, TCP utilise la technique du *three-way handshake. Dans* le cas le plus simple, cette technique procède comme il est illustré dans la figure ci-dessous:



Séquence de messages dans un *three way acknoledgement*.

Les segments SYN transporte les informations sur le numéro de séquence initial.

La technique du *three-way handshake* accomplit deux fonctions importantes :

- Elle garantit que les deux extrémités sont prêtes à transférer les données (et elles savent qu'elles sont toutes les deux prêtes).
- Elle leur permet de s'accorder sur les numéros initiaux des séquences. Les numéros de séquences sont transmis et reconnus pendant le *Three-way handshake*. Chaque machine choisit un numéro de séquence initial qui va être utilisé pour identifier les bytes dans le flot de données qu'elle va transmettre.

Il est important que les deux extrémités s'accordent sur le numéro initial pour que les numéros de bytes utilisés dans les accusés de réception s'accordent avec ceux utilisés dans les segments de données. Les machines s'accordent sur les numéros de séquences pour deux flots de données après seulement trois messages, puisque chaque segment contient à la fois un champ du numéro de séquence et un champ de l'accusé de réception.

Le premier segment d'un *Three-way handshake* peut être identifié car il a le bit SYN dans le champ code à 1. Le deuxième message a à la fois le bit SYN et les bits ACK, indiquant qu'il accuse réception du premier segment SYN et qu'il continue le *handshake*. Le dernier message est seulement un accusé de réception et est simplement utilisé pour informer la destination que les deux côtés sont d'accord sur l'établissement de la connexion. Généralement, le logiciel de TCP sur une machine attend passivement le *Three-way handshake*, sur l'autre machine il l'amorce.

Le Three-way handshake est conçu pour bien fonctionner même si les deux machines tentent d'établir une connexion simultanément. Une fois la connexion établie, les données peuvent circuler dans les deux sens. Comme les messages transmis peuvent se perdre ou arriver en retard, le protocole doit utiliser les mécanismes de délais d'attente et de retransmission des requêtes perdues.

D'autres problèmes peuvent, également, apparaître quand les demandes de retransmission arrivent pendant l'établissement de la connexion, ou quand elles sont retardées jusqu'à ce qu'une connexion soit établie, utilisée et terminée. La technique du *three-way handshake* résout ces problèmes.

La machine A, qui se trouve dans le site 1, qui amorce le *handshake*, passe son numéro de séquence initial x, dans le premier champ de séquence SYN du premier segment du *threeway handshake*. La deuxième machine B, qui se trouve dans le site 2, reçoit le SYN, enregistre le numéro de séquence et répond en envoyant son numéro de séquence initial dans le champ séquence ainsi qu'un accusé de réception qui spécifie que B attend le byte x + 1.

Dans le dernier message du *handshake*, A reconnaît tous les bytes jusqu'au numéro y qu'elle reçoit de B. Dans tous les cas, des accusés de réception suivent la convention en utilisant le prochain numéro de byte attendu. Avec une telle conception du protocole, il est possible de transmettre des données avec les numéros de séquence initiaux dans les segments du *handshake*. Pour cela, le logiciel du protocole doit garder les données jusqu'à la terminaison *du handshake*. Une fois la connexion est établie, le logiciel de TCP peut libérer ces données et les livrer rapidement au programme d'application qui les attend.

2.4.1.12 Fermeture d'une connexion

Les connexions TCP sont *Full Duplex* et donc elles peuvent être vues comme si elles contenaient deux transferts indépendants de données, dans les deux sens. Quand un programme d'application dit à TCP qu'il n'a plus de données à transférer, TCP ferme la connexion dans ce sens. Pour fermer l'autre moitié de la connexion, l'émetteur termine la transmission des données restantes et envoie un segment avec le bit FIN mis à 1.

Le TCP récepteur accuse réception du segment FIN et informe le programme d'application qu'il n'y a plus de données disponibles. Une fois la connexion fermée dans un des deux sens, TCP refuse d'accepter des données émises dans ce sens. Pendant ce temps, les données peuvent continuer à circuler dans le sens opposé jusqu'à sa fermeture par l'émetteur. Quand les connexions sont fermées dans les deux sens, les données ne circulent plus et la connexion est supprimée.

2.4.1.13 Remise à zéro de la connexion

Un programme d'application ferme une connexion quand il a fini de l'utiliser. Ainsi, la fermeture est considérée comme une utilisation normale. Parfois, des conditions anormales apparaissent et force un programme d'application ou le logiciel

du réseau de rompre la connexion. TCP offre une opération de remise à zéro pour de telles déconnexions anormales.

Une extrémité de la connexion amorce une remise à zéro en envoyant un segment avec le bit RST dans le champ CODE mis à 1. L'autre extrémité répond immédiatement à une remise à zéro en faisant échouer la connexion. Elle informe également le programme d'application qu'une remise à zéro vient de se produire. Un avortement signifie que les transferts dans les deux sens cessent immédiatement, et les ressources telles que les mémoires-tampon seront libérées.

2.4.1.14 Livraison forcée

TCP peut diviser le flot de données en segments pour les transmettre sans tenir compte des tailles de transfert utilisées par les programmes d'application. L'avantage principal, permettant à TCP de choisir une division, est l'efficacité. Il peut accumuler plusieurs bytes dans une mémoire-tampon pour rendre les segments assez longs, réduisant les frais qui se présentent quand les segments contiennent seulement peu de bytes de données.

Malgré que la mémoire-tampon améliore le débit du réseau, elle peut déranger quelques applications. Si on utilise une connexion TCP pour passer des caractères entre un terminal interactif et une machine distante, l'utilisateur s'attendrait à une réponse instantanée après chaque frappe au clavier. Si TCP met les données dans ses mémoires-tampon, la réponse pourrait être retardée.

TCP fournit une opération *push* qui peut être utilisée par un programme d'application pour forcer la livraison des données actuellement dans le flot de données sans attendre que la mémoire-tampon les remplisse. L'opération *push* fait, plus que forcer TCP à transmettre un segment. Elle demande également à TCP de mettre à 1 le bit PSH dans le champ code du segment, de façon à ce que les données soient livrées au programme d'application récepteur.

Ainsi, quand on envoie des données d'un terminal interactif, l'application utilise la fonction *push* après chaque frappe au clavier. De façon similaire, les programmes d'application peuvent envoyer et éditer rapidement la sortie standard sur le terminal en faisant appel à la fonction *push* après avoir écrit un caractère ou une ligne.

En plus de la fonction *push,* TCP fournit un mécanisme appelé URGENT POINTER qui permet à l'émetteur d'informer le récepteur qu'une donnée urgente va arriver et doit être traitée rapidement. Par exemple, dans une connexion TCP utilisée pour un trafic entre un terminal interactif et un ordinateur, les caractères qui arrêtent et qui relancent la sortie standard (contrôle-s et contrôle-q) peuvent être considérés comme urgents puisque le récepteur doit les traiter immédiatement.

2.4.1.15 Numéros de port réservés

TCP combine les liaisons d'accès statiques et dynamiques de la même façon que UDP, en utilisant un ensemble de numéros d'accès ou numéros de ports (courrier

électronique etc.). Il laisse la majorité de ces numéros disponibles dans le système d'exploitation pour que les programmes d'application puissent les allouer en cas de besoin. Le tableau ci-dessous illustre les accès ou numéros de ports TCP affectés actuellement (un tableau semblable mais plus détaillé est présenté en annexes) :

Numéro de port	Nom	Description
0		Réservé
1-4		Non assigné
5	RJE	Entrée à distance des travaux
7	ECHO	Echo
9	DISCARD	Abandonner
11	USERS	Utilisateurs actifs
13	DAYTIME	Journée
15	NETSTAT	Statistiques du réseau
17	QUOTE	Numéro de référence du jour
19	CHARGEN	Générateur de caractères
20	FTP-DATA	Protocole de transfert de fichiers (donnée)
21	FTP	Protocole de transfert de fichiers
23	TELNET	Connexion du terminal
25	SMTP	Protocole simple de transport du courrier
37	TIME	L'heure
39	RLP	Protocole de localisation des ressources
42	NAMESERVER	Serveur des noms des hôtes
43	NICNAME	Who Is
53	DOMAIN	Serveur du nom du domaine
67	BOOTPS	Serveur du protocole Bootstrap
68	BOOTPC	Client du protocole Bootstrap
69	TFTP	Transfert trivial de fichiers
75		Service privé d'appel téléphonique
77		Service privé RJE
79	FINGER	Finger
95	SUPDUP	Protocole SUPDUP
101	HOSTNAME	Serveur des noms des hôtes NIC
102	ISO-TSAP	ISO-TSAP
113	AUTH	Service d'authentification
117	UUCP-PATH	Service des chemins UUCP
123	NTP	Protocole du temps réseau
133-159		Non assigné
160-223		Réservé
224-241		Non assigné
247-255		Non assigné

Malgré que les accès TCP et UDP sont indépendant, les concepteurs ont choisi d'utiliser les mêmes numéros de ports pour les services accessibles à la fois par UDP et TCP.

2.4.2 Protocole UDP

2.4.2.1 Introduction

Le protocole UDP *(User Data Protocol)* est un protocole de transport qui se situe au dessus d'IP. Il assure un transport non fiable et sans connexion. C'est un protocole qui permet d'envoyer des trames qui contiennent l'adresse IP de la machine émettrice et celle de la machine de destination. Le protocole UDP utilise des numéros de ports pour spécifier une application sur une machine.

UDP est un protocole standard d'Internet, il permet à un programme d'application sur une machine d'envoyer un datagramme à un programme d'application sur une autre machine. Il utilise le protocole Internet pour livrer les datagrammes. D'une façon générale, la seule différence entre UDP et IP est que le message UDP comprends des numéros de port, permettant à l'émetteur de faire une distinction entre plusieurs destinations (programmes d'application) sur la machine cible. UDP comprend également une somme de contrôle de la donnée transmise.

UDP fait ainsi une distinction entre plusieurs processus sur une machine donnée en permettant aux émetteurs et récepteurs d'ajouter à chaque message UDP deux entiers de 16 bits spécifiant les numéros de port. Ces numéros identifient la source et la destination. Quelques numéros de port UDP sont réservés (par exemple: port 69 est réservé pour le protocole Internet TFTP : *Trivial File Transfer Protocol*). D'autres numéros de port sont disponibles pour l'utilisation d'autres programmes d'application.

Dans la structure en couches, UDP est un des protocoles qui se situe au dessus de la couche IP. Ces deux couches sont indépendantes, cependant il y a une forte interaction entre les deux. UDP est un mécanisme qui fournit aux processus utilisateurs un accès au service de livraison de paquets d'Internet qui est non fiable et sans connexion.

2.4.2.2 Les ports

La plupart des ordinateurs modernes sont multitâches, ils permettent l'exécution simultanée de plusieurs programmes d'application. Chaque programme en exécution est un processus (processus niveau utilisateur). Les processus sont créés, puis détruits de façon dynamique. Seuls les émetteurs connaissent les informations nécessaires pour identifier un processus sur une machine cible. Chaque machine contient un ensemble de points de destination indépendants appelés les ports du protocole. Typiquement, chaque port du protocole est identifié par un entier positif. Le système d'exploitation local choisit un mécanisme que les processus utilisent pour spécifier un port ou pour y accéder.

2.4.2.3 Structure du protocole

Dans le protocole TCP/IP, UDP (*User Datagram Protocol)* fourni un mécanisme que l'émetteur utilise pour distinguer une application parmi plusieurs programmes d'application récepteurs sur une même machine. En plus des données transmises par un processus utilisateur, chaque message UDP contient à la fois un numéro de port de destination et un numéro de port de source, permettant ainsi au logiciel UDP de livrer le message au récepteur correspondant, et à ces derniers d'envoyer une réponse.

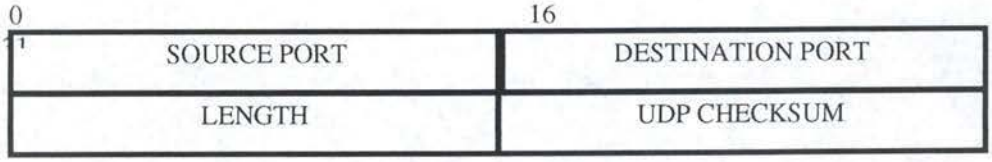
Comme UDP dépend du protocole Internet utilisé pour transporter un message d'une machine à une autre, il fournit le même service qu'IP, c'est-à-dire un service de livraison non fiable et sans connexion. Il n'utilise pas des accusés de réception pour s'assurer de l'arrivée de messages, il n'ordonne pas les messages reçus et il ne fourni pas un *feedback* pour le contrôle du taux de transmission des données entre les deux machines. Ainsi, les messages UDP peuvent être perdus, dupliqués ou arrivés en

désordre. De plus, les paquets peuvent arriver à la machine de destination à une vitesse supérieure à celle de leur traitement ce qui peut entraîner des risques de surcharge.

UDP fournit ainsi un service de livraison non fiable et sans connexion utilisant IP pour le transport des messages entre les machines. Il ajoute des informations pour préciser une destination spécifique sur un ordinateur hôte donné.

2.4.2.4 Format UDP

Chaque message UDP est appelé un datagramme utilisateur (USER DATAGRAM), il se compose de deux parties: l'en-tête et la zone de données. L'en-tête est divisé en 4 champs de 16 bits qui spécifient le port source, le port de destination, la longueur du message et une somme de contrôle. Les détails des champs de l'en-tête sont illustrés dans ce qui suit:



Format des champs dans l'en-tête UDP

Les champs SOURCE PORT et DESTINATION PORT contiennent des numéros de port de 16 bits utilisés pour démultiplexer les datagrammes sur les processus récepteurs. Le port source est optionnel. Il est utilisé pour spécifier le port auquel les réponses doivent être transmises; s'il n'est pas utilisé, sa valeur serait nulle.

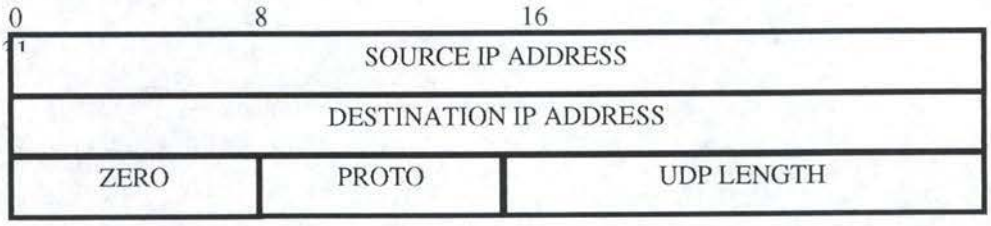
Le champ longueur (LENGTH) contient le nombre d'octets du datagramme, incluant l'en-tête et les données utilisateurs. Ainsi, la valeur minimale de ce champ est égale à 8, la longueur de l'en-tête.

Le champ somme de contrôle UDP est également optionnel; la valeur zéro signifie que la somme n'a pas été calculé. Pour calculer la somme de contrôle, UDP crée un pseudo en-tête, ajoute comme suffixe un octet de zéros pour remplir le datagramme à exactement un multiple de 16 bits, et calcule la somme de contrôle sur la totalité de ce nouveau datagramme. Cet octet de zéros est ajouté à l'en-tête seulement.

L'octet utilisé pour le remplissage et le pseudo en-tête ne sont pas transmis avec le datagramme UDP, ils ne sont pas non plus inclus dans la longueur. UDP utilise le même algorithme de somme de contrôle qu'IP. Il traite le pseudo en-tête, l'en-tête, la donnée et l'octet de remplissage comme une séquence d'entiers de 16 bits. Pour calculer une somme de contrôle, le logiciel accumule d'abord une somme en complément à un de 16 bits, il prend ensuite les 16 bits complémentaires de la somme pour produire la somme de contrôle. Pendant le calcul de la somme de contrôle, le champ de celui-ci est supposé contenir zéro.

L'en-tête UDP spécifie seulement le numéro de port du protocole. Ainsi, pour vérifier la destination, UDP sur la machine émettrice calcule une somme de contrôle qui couvre l'adresse hôte de destination et le datagramme UDP. A la dernière destination, le logiciel UDP vérifie la somme de contrôle en utilisant l'adresse Internet de destination obtenue de l'en-tête du datagramme IP qui transporte le message UDP. Si la somme de contrôle correspond alors le datagramme a correctement atteint la destination et le port désirés.

Le pseudo en-tête UDP a une longueur de 12 octets arrangée comme suit:



Bytes du pseudo en-tête utilisés pour le calcul de la somme de contrôle.

Les champs SOURCE IP ADDRESS et DESTINATION IP ADDRESS du pseudo en-tête contiennent les adresses Internet source et destination qui vont être utilisés pour la transmission des messages UDP. Le champ proto contient le type de protocole IP sous forme d'un code, ce code est égal à 17 pour UDP, le champ LENGTH UDP contient la longueur du datagramme UDP.

Pour vérifier la somme de contrôle, le récepteur doit extraire ces champs de l'en-tête IP, il les assemble en un format pseudo en-tête et ensuite, il recalcule la somme de contrôle après avoir ajouté l'octet de zéros. Si le résultat correspond au champ UDP CHECKSUM, le paquet est intact.

2.4.2.5 UDP et les couches de protocole

La couche UDP se situe, comme on l'a déjà indiqué, au dessus de la couche IP. Les messages UDP complets (en-tête et donnée) sont encapsulés dans la zone de données du datagramme IP avant de traverser l'Internet.

Quand on considère la façon dont les en-têtes sont insérés et enlevés, il est important de penser au principe des couches. En particulier, ce principe s'applique à UDP, ainsi le datagramme UDP reçu dans la couche IP sur la machine de destination est identique au datagramme que UDP passe à IP sur la machine source. Les données qu'UDP livre à un processus utilisateurs sur la machine réceptrice seront exactement les mêmes qu'un processus utilisateur passe à UDP sur la machine émettrice.

La couche IP est seulement responsable des transferts des données entre les hôtes sur Internet alors que la couche UDP est responsable seulement de la différentiation entre les multiples sources ou destinations dans le même hôte. Ainsi, seul l'en-tête IP qui peut identifier les hôtes source et destination; et seule la couche UDP qui peut identifier les ports source ou destination dans un hôte.

2.4.2.6 Calcul de la somme de contrôle et le modèle en couche

L'adresse source IP dépend de la route choisie pour le datagramme, car l'adresse source identifie l'interface réseau sur laquelle le datagramme est transmis. Ainsi, UDP ne peut pas connaître une adresse source IP à moins qu'il interagisse avec la couche IP.

On suppose que le logiciel UDP demande à la couche IP de calculer les adresses IP, sources et destinations, les utilise pour construire un pseudo en-tête, calcule la somme de contrôle, enlève le pseudo en-tête et passe le datagramme UDP à IP pour la transmission.

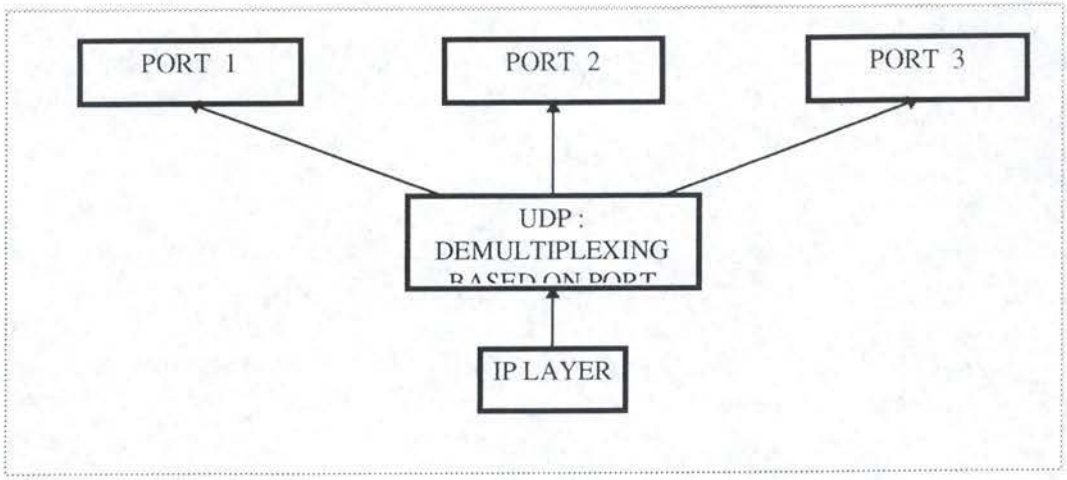
Une autre approche plus efficace peut être définie:

- encapsuler les messages UDP dans les datagrammes IP (dans la couche UDP),
- remplir les champs source et destination IP de l'en-tête,
- calculer la somme de contrôle UDP sur l'en-tête, et
- passer le datagramme IP à la couche IP. Celle-ci va remplir les champs restants de l'en-tête IP.

2.4.2.7 Démultiplexage UDP

Le terme multiplexage utilisé dans ce modèle est différent de celui du modèle OSI. Il s'agit en fait d'un routage au niveau de l'adressage interne.

Dans les couches de la hiérarchie du protocole, le logiciel doit multiplexer ou démultiplexer plusieurs choses pour la couche suivante. Le logiciel UDP est un exemple de multiplexage. Il accepte les datagrammes UDP du software IP et les démultiplexe. Un exemple de démultiplexage dans la couche UDP est illustré dans la figure ci-dessous:



Quand un datagramme UDP arrive, il est routé selon son numéro de port. C'est ce qu'on appelle un démultiplexage basé sur le port.

Les messages UDP utilisent le numéro de port pour sélectionner une destination appropriée des datagrammes reçus.

2.5 Protocoles de niveau Application.

Les applications réseaux disponibles sur TCP-UDP/IP peuvent être classées en trois groupes :

applications de base, applications étendues, applications Unix.

2.5.1 Applications de base

Il s'agit d'applications existant sur tout système supportant Internet, quels que soient le système d'exploitation et le réseau utilisés. Ce sont les applications de base des réseaux hétérogènes. Il faut toutefois noter que si le système d'exploitation n'est pas multi-tâche, seuls les clients seront installés, les serveurs correspondant ne pouvant l'être. Il s'agit de :

2.5.1.1 TFTP (Trivial File Transfer Protocol) (UDP).

Transfert de fichiers en mode datagram. Utilisée par les terminaux X et les stations de travail sans disque lors de la phase de démarrage (*boot* sur le réseau), l'application TFTP pourrait être utilisée pour des transferts de fichiers dans le cas général.

Cette commande peut poser des problèmes de sécurité : aucun mécanisme d'identification n'est mis en oeuvre. Il est possible de sécuriser le *serveur* TFTP d'une machine Unix, en restreignant ses possibilités d'accès à une partie de l'arborescence de fichiers. Dans la plupart des cas, si la fonctionnalité *serveur de boot* pour des terminaux X ou des stations sans disque n'est pas nécessaire sur une machine, il est préférable de ne pas installer le *serveur TFTP* sur cette machine (pour raisons de sécurité).

2.5.1.2 FTP (File Transfer Protocol) (TCP).

Transfert de fichiers en mode connecté. Cette commande négocie une connexion avec identification de l'utilisateur sur la machine distante : si un utilisateur A sur la machine *alpha* échange par *FTP* des fichiers avec un utilisateur B sur la machine *beta*, cet utilisateur devra connaître le *login* et le mot de passe de B sur *beta*. Pour des raisons de sécurité évidentes, ce ne peut être que la même personne (pouvant avoir des noms de *login* différents sur *alpha et beta).*

Il existe une exception très intéressante à cette règle : le **FTP anonyme (anonymous FTP),** permettant à l'utilisateur A sur *alpha* de lire (jamais d'écrire) des

fichiers se trouvant sur la machine *beta* sans identification préalable. Cela suppose, de la part de l'administrateur de *beta,* un travail de configuration supplémentaire du *serveur FTP* pour que cette connexion anonyme ne pose pas de problèmes de sécurité.

Cette possibilité est très répandue sur le réseau Intemet : certains centres informatiques mettent ainsi à la disposition de la communauté Internet des logiciels dits « du domaine public », des informations sur l'état de leurs programmes de recherche, etc.

Le client FTP est lancé par la commande

```
ftp [option] [nom_ de_ machine]
```

Les options ne seront pas explicitées ici (utilisez *man ftp* pour en savoir plus). Le champ *nom_de_machine* est une adresse Internet absolue ou symbolique, par exemple 138.48.32.99 ou babbage@info.fundp.ac.be. Si ce champ est omis, le *client TFTP* passe immédiatement en mode commande. Dans le cas où le champ *nom_de_machine* est spécifié, la connexion est établie, l'utilisateur s'identifie sur la machine distante, puis FTP passe en mode commande (le prompt est ftp>).

Attention
FTP établit une connexion avec identification sur le site distant. mais il ne s'agit pas d'un *login* (une session de travail). Pour ce faire. il faut utiliser *TELNET.*

2.5.1.3 TELNET (TErminaL NETwork protocol).

Emulation de terminal sur le réseau, permet la connexion à distance. Tout ce qui a été dit sur la connexion et l'identification sur le site distant à propos de *FTP* est applicable *à TELNET* (sauf, bien sûr, la connexion anonyme!).

Le client TELNET est lancé par la commande

```
telnet [nom_de_machine]
```

Le champ *nom de machine* est une adresse Intemet absolue ou svmbolique, par exemple 138.48.32.99. Il ou babbage@info.undp.ac.be. Si ce champ est omis, le client TELNET passe immédiatement en mode commande (le prompt est *telnet>).* Dans le cas où le champ *nom_de_machine* est spécifié, la connexion est établie, l'utilisateur s'identifie sur la machine distante et entre en session sur cette machine.

Il est possible de passer en mode commande par un mécanisme d'échappement (<ctrl-t>).

Nous n'irons pas plus loin dans la description de *TELNET.* La commande Unix *man telnet* (ou la commande *help de TELNET)* permet d'en savoir plus sur les possibilités de son implémentation de cette application très simple d'utilisation.

2.5.1.4 SMTP (Simple Mail Transfer Protocol).

Utilisée pour l'échange de messages ascii par courrier électronique, cette application n'est pas utilisée directement, mais par l'intermédiaire de la commande mail (UNIX).

2.5.2 Applications étendues

Ces applications, de haut niveau, ne font pas partie de TCP-UDP/IP, mais ont été développées sur ces protocoles. Elles ne sont pas limitées au monde Unix, et sont disponibles sur d'autres systèmes d'exploitation.

Elles ont largement contribué au développement des réseaux hétérogènes. Il s'agit de :

2.5.2.1 RPC Remote Procedure Call (UDP).

Ce protocole permet l'exécution de procédures sur une machine distante. Equivalent à distance d'un appel conventionnel de procédure, ce mécanisme est dit synchrone : le processus en cours attend la fin de l'exécution de la procédure distante pour continuer. Ce protocole est utilisé par des applications de plus haut niveau, en particulier par *NFS*.

2.5.2.2 NFS Network File System. (UDP), RFS Remote File Sharing (TCP).

Ces deux applications sont fonctionnellement équivalentes. NFS, développée par Sun, est universellement utilisée au point d'être le standard de fait. RFS d'ATT est sur TCP, ce qui permet une meilleure sécurité d'accès, mais au prix d'une dégradation importante des performances.

Ce type d'application permet de greffer une partie de l'arborescence de fichiers d'une machine *alpha* en un point de l'arborescence d'une machine *beta* . La machine *alpha* «exporte» un répertoire, la machine *beta le* monte sur un répertoire de sa propre arborescence. (Cette opération de montage peut être effectuée par plus d'une machine.)

Exemple

La machine *alpha* exporte le répertoire */usr/share/lib.* La machine *beta* monte ce répertoire sur son répertoire local */usr/lib.* Le fichie */usr/share/lib/naglib.a de alpha* est accessible sous le nom *I/usr/lib/naglib.a* sur la machine *beta,* et rien ne le différencie d'un fichier local.

Les opérations d'export et de montage sont effectuées par l'administrateur, qui peut restreindre les droits d'accès aux répertoires exportés. Ce point ne sera pas détaillé ici.

2.5.2.3 X Window (TCP).

C'est une application réseau développée sur TCP. Lorsque X Window est utilisée entièrement sur une machine, les clients X locaux accèdent au serveur X local de la même façon qu'ils le feraient pour un serveur X distant. (IP définit une adresse IP particulière 138.48..1 localhost : la machine elle-même.)

2.5.3 Applications Unix

Variantes à distance de commandes Unix standard, les *remote commands* sont des commandes habituelles précédées de la lettre **r** (pour *remote).* Toutes ces commandes nécessitent évidemment l'identification de l'utilisateur sur la machine distante. Pour toutes les *remote commands,* cette identification repose sur des fichiers de configuration réseau décrits plus loin.

2.5.3.1 rlogin (Remote LOGIN).

```
rlogin nom_de_machine [ -l nom_login]
```

Cette application est fonctionnellement équivalente à *telnet,* mais de machine Unix à machine Unix. L'option *-l nom_login* permet de spécifier sous quel nom l'utilisateur se connecte à la machine distante; sinon le nom local est utilisé.

rlogin présente par rapport *à telnet* deux avantages : l'environnement Unix est transmis à la machine distante, et l'identification sur la machine distante peut être automatisée sans que la sécurité en soit gravement affectée (au contraire). En pratique, pour des raisons trop longues à détailler, l'éditeur *vi* fonctionnera souvent mieux si la connexion est établie par *rlogin* plutôt que par *telnet.*

L'identification n'est pas automatisée (rlogin est la seule « *remote command* » pouvant fonctionner sans identification automatique).

2.5.3.2 rcp (Remote CoPy).

rcp permet la copie de fichiers entre la machine locale et une machine distante, ou entre deux machines distantes.

```
rcp source destination
rcp  -r   répertoire_source   répertoire_destination
```

Cette extension de la commande *cp* nécessite, pour la désignation des noms de fichiers ou de répertoires à copier, la définition de la machine sur laquelle ils se trouvent :

```
utilisateur@nom_de_machine :nom_de_fichier
```

Le champ utilisateur @ peut être omis si le nom de l'utilisateur est le même sur les deux machines.

2.5.3.3 rsh (Remote shell).

rsh permet de lancer une commande Unix sur une machine distante.

```
rsh nom_de_machine [ -l  nom_login] [commande]
```

L'option *-l nom login* a le même sens que pour *rlogin*. Si la commande est omise, *rsh* est équivalent à *rlogin*.

2.5.4 Fichiers de configuration.

2.5.4.1 Fichiers de configuration globale

Ces fichiers concernent l'administrateur système et nous n'entrerons pas dans le détail de leur description. Signalons simplement l'existence des fichiers suivants :

/ect/host

Ce fichier contient une liste de couples (numéro IP, nom symboliques). C'est par son intermédiaire que IP transforme un nom symbolique de machine en une adresse IP.

/ect/resolve.conf

Dans un grand réseau, il serait fastidieux (voire impossible) de garantir que le fichier */ect/hosts est* à jour sur chaque machine. Il est alors possible de définir un « serveur de noms » (DNS : Domain Name Server), sur lequel une base de données fonctionnellement équivalente à un fichier */etc/hosts* est soigneusement maintenue à jour. Le fichier */ect/resolve.conf* décrit la liste des serveurs de noms disponibles sur le domaine et leurs adresses IP.

/etc/hosts.equiv

Ce fichier permet de déclarer que plusieurs machines du réseau sont équivalentes. Des utilisateurs ayant des noms identiques sur des machines déclarées équivalentes n'ont pas besoin de s'identifier lors d'une session réseau. C'est une facilité, mais également un trou potentiel de sécurité (à n'utiliser qu'avec une extrême prudence).

/etc/networks

Ce fichier est aux réseaux locaux ce que le fichier */etc/hosts* est aux machines : il permet d'établir une relation entre noms symboliques de réseaux et numéros IP de ces réseaux.

/etc/services

Ce fichier contient la liste des services réseaux éventuellement disponibles sur cette machine nom - numéro de port - protocole.

2.5.4.2 Fichiers de configuration personnelle

Ces fichiers sont créés par l'utilisateur pour configurer son propre environnement réseau.

$HOME/.rhosts

Sur la machine *yoda* le fichier /user/sdandoy/.rhosts
babbage sdandoy

L'utilisateur *sdandoy de yoda* déclare faire confiance à l'utilisateur *sdandoy* de babbage.

Dans ce cas, une *remote command* lancée sur la machine yoda *par* l'utilisateur sdandoy de la machine *babbage* sera exécutée sans identification préalable. Il est bien évident que l'utilisateur *de babbage* est la même personne que l'utilisateur sdandoy de yoda.

$HOME/.netrc

Ce fichier est l'équivalent du précédent pour *ftp et telnet;* il contient votre mot de passe en clair! **NE L'UTILISEZ JAMAIS**. L'administrateur de votre système devrait prendre soin de supprimer tout fichier *.netrc exis*tant sur la machine dont il a la responsabilité.

Deuxième Partie : Eléments de la Sécurité.

1 Introduction

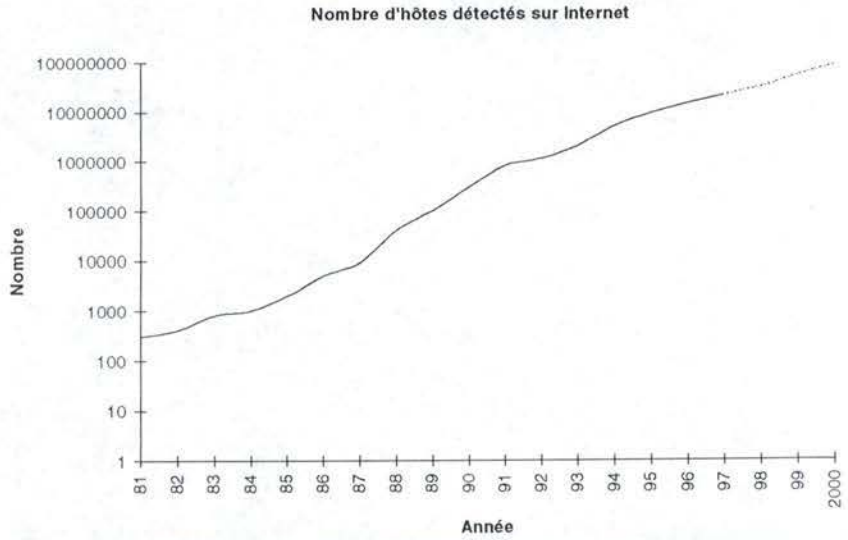
1.1 Définition et origine du problème

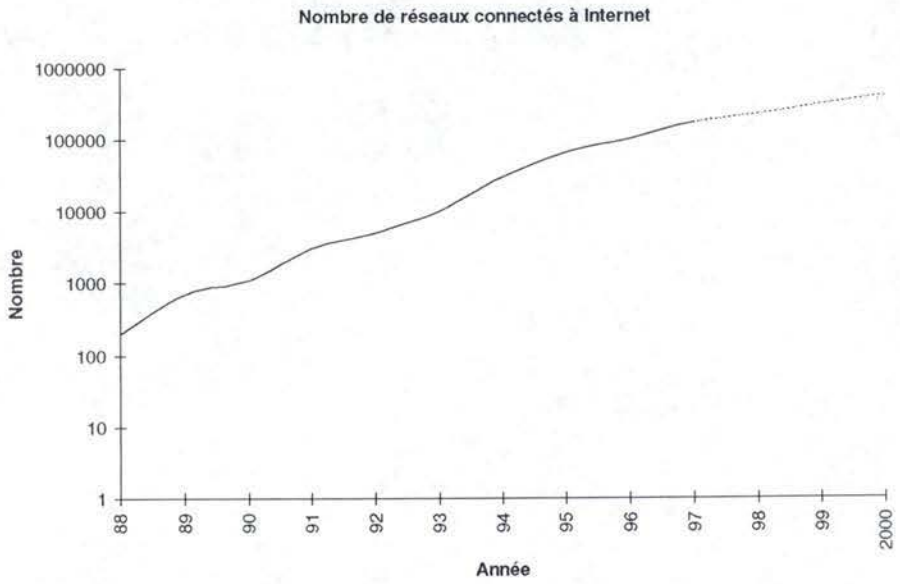
Définition encyclopédique de la sécurité : situation, état tranquille qui résulte de l'absence de danger.

Alors que les ordinateurs deviennent de plus en plus nombreux, de plus en plus petits et de moins en moins chers, les gens sont devenus de plus en plus intéressés par les moyens leur permettant de connecter leur ordinateurs ensemble pour former des réseaux ou des systèmes distribués.

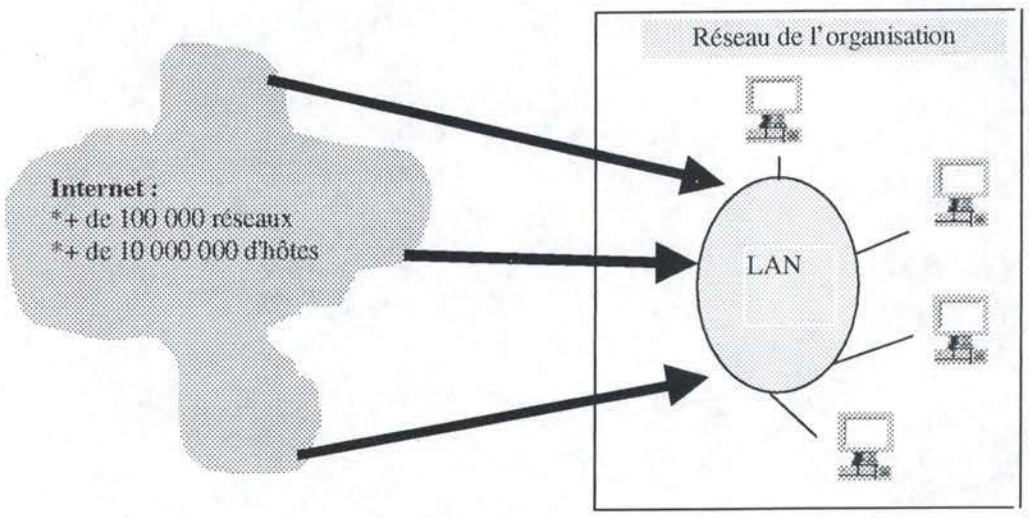
Concrètement, une connexion à Internet permet d'étendre les dimensions de son réseau aux dimensions de la planète. En effet, Internet c'est 10 000 000 d'hôtes (et donc combien de personne ?), c'est 100 000 réseaux.

Sur les deux graphiques ci-dessous sont repris l'évolution du nombre d'hôtes et du nombre de réseaux aux cours de ces dernières années. Comme on peut le voir, l'échelle est semi-logarithmique ; l'accroissement est donc exponentiel. La sécurité est un phénomène qui ne peut pas aller en s'améliorant et auquel il faudra accorder tôt ou tard son attention.





Grâce à Internet, on se trouve donc connecté, par l'intermédiaire d'**une ou de plusieurs connexions**, à 100 000 autres réseaux et on a la possibilité de converser avec plus de 10 000 000 d'autres utilisateurs. Toutes ces connexions constituent autant de brèches dans le **périmètre de sécurité**. Une connexion à Internet, c'est la porte ouverte à l'espionnage industriel, au sabotage, aux accès non autorisés, etc.



Si on adapte la définition générale au domaine des télécommunications on obtient : la sécurité c'est empêcher quiconque de faire ce qu'on n'a pas envie qu'il fasse à l'encontre de nos machines, avec nos machines ou à partir de nos machines.

1.2 Les qualités recherchées.

Le but de la sécurité sur Internet est de **colmater ces brèches** et donc **rétablir un périmètre de sécurité fort** de manière à permettre aux utilisateurs de l'organisation d'accéder à Internet tout en empêchant les utilisateurs extérieurs non-autorisés à accéder aux ressources de l'entreprise. Tout cela va dépendre de la **politique de sécurité** adoptée dans l'organisation.

Le but de la sécurité est en fait de conserver les trois qualités fondamentales suivantes :

- **la confidentialité**: qui consiste à protéger l'information et la garder secrète;
- **l'intégrité**: qui consiste à conserver l'information et la garder intacte. Ainsi, elle ne subit de modifications que par l'intervention des utilisateurs autorisés;
- **la disponibilité**: qui consiste au bon fonctionnement du système. Ainsi les ressources et les informations doivent être accessibles dans des délais convenables par les utilisateurs autorisés.

1.3 Les différents types de solutions.

Plusieurs mesures peuvent annihiler les attaques et violations de la sécurité des systèmes informatiques. Il y a les contrôles d'accès aux systèmes, aux machines ainsi qu'aux terminaux. Et d'autres, qui sont plus spécifiques, telles que:

- **les contrôles d'accès aux données et aux informations sensibles**. Il y a deux types de contrôles d'accès:

  contrôle d'accès discrétionnaire (DAC: *Discretionnary Access Control*): dans ce cas, l'utilisateur est le seul responsable, c'est lui qui choisit le type de protection de ses fichiers[1];

  contrôle d'accès mandaté (MAC: *Mandatory Access Control*): ici, les systèmes se chargent de la protection des fichiers. Ce mode de contrôle est adopté par des ordinateurs qui manipulent des données très sensibles;

- **la cryptographie**: elle consiste à chiffrer les données au moyen d'un jeu de clés, de manière à les rendre illisibles à celui qui ne possède pas la clé. En général, le chiffrement est effectué par des algorithmes de codification; un algorithme inverse, permettra ensuite de reconstituer l'information originale:, Parmi les algorithmes les plus répandus en cryptographie, on peut citer l'algorithme DES (Data Encryption Standard), RSA (Revest, Shamir and Adelmann);

- d'autres méthodes de sécurité sont utilisées telles que les **sauvegardes**, les **procédures antiviraux**, etc.

- Les ***firewalls*** : On définit un *Firewall* comme une liste de composants placés entre deux réseaux et qui respectent tous certaines propriétés :

[1] C'est un peu le cas des facultés, où tout le monde est responsable des données qu'il laisse sur son disque.

- Tout trafic de l'intérieur vers l'extérieur, et inversement, doit passer à travers le *Firewall*.
- Seul le trafic autorisé d'après la politique de sécurité est autorisé à passer.
- Le *Firewall*, lui-même, doit être impénétrable.

•**l'audit de la sécurité**: cette technique consiste à enregistrer dans un ou plusieurs fichiers des informations captées, sur le réseau, sur l'activité des utilisateurs.

Ce sont ces deux techniques qui vont être employées dans ce mémoire. Il va en fait s'agir de faire tourner sur une machine correspondant à la définition de ***firewall*** un logiciel d'**audit** s'appelant ASAX : Advanced Security Audit Trail Analysis on Unix.

Dans cette partie, nous allons tout d'abord voir les différents types d'attaques qui existent pour les différents protocoles vus dans la première partie, nous verrons ensuite ce qu'est un firewall et quel type de firewall serait le plus adapté à notre type de situation. Après cela, nous verrons comment fonctionne ASAX. C'est la troisième partie de ce mémoire qui sera consacrée à l'implémentation et à la réalisation concrète d'un outil de sécurité pour les facultés.

2 Les attaques réseaux.

2.1 Introduction.

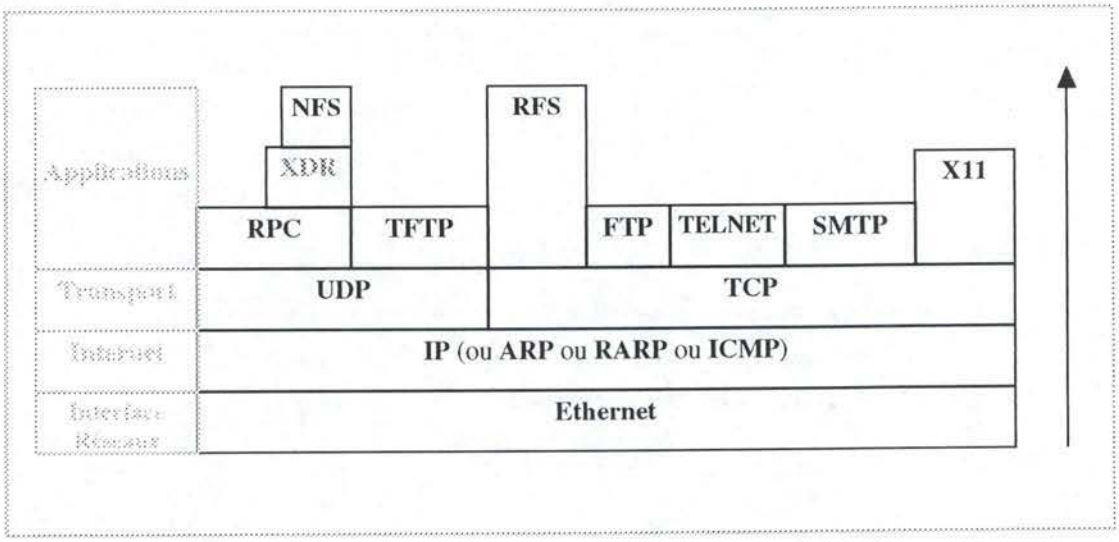
Le protocole TCP/IP, très utilisé de nos jours, a été développé sous l'égide du DOD (Department Of Defense). Malgré cela, il subsiste des imperfections inhérentes à ces protocoles[2]. Certains défauts sont dus au fait que l'authentification d'un hôte repose sur l'adresse IP source (p.e. : les « r-utilities » de Berkeley). D'autres défauts se situent au niveau des mécanismes de contrôle des réseaux et, en particulier, des protocoles de routage qui possèdent un processus d'authentification minimal voire inexistant.

Cette section présente les différentes attaques basées sur ces imperfections. **Ces différentes attaques reposent sur l'hypothèse que le pirate possède un contrôle plus ou moins complet sur une machine connectée au réseau Internet.** Cela peut être dû à des défauts dans les mécanismes de protection de cette machine, ou parce que cette machine est un micro-ordinnateur (non-protégé par définition). On peut aussi avoir affaire à un administrateur système espiègle. Notons également que dans le terme « pirate » il faut sous-entendre non seulement sabotage, mais aussi espionnage, falsification d'informations, etc.

[2] Ces défauts n'ont rien à voir avec la qualité de l'implémentation.

On pourra également trouver, dans cette section, des idées de défenses possibles contre ces attaques.

Pour bien comprendre les mécanismes de ces différentes attaques, il est conseillé de se reporter à la première partie de ce mémoire, celle concernant la description du fonctionnement des différents protocoles. Comme cela avait été fait dans cette partie, les attaques vont également être présentées suivant un ordre croissant de leur numéro dans le modèle Internet (tel que présenté sur la figure ci-dessous) .



2.2 Problèmes de sécurité avec la suite de protocoles TCP/IP.

2.2.1 Protocole de niveau Interface réseau : Ethernet.

Ethernet
Rejet de service[3]

Il s'agit d'exploiter le principe de CSMA/CD pour réaliser une attaque de rejet de service. Le principe de base est généralement de provoquer des collisions à répétitions sur le support Ethernet. Pour ce faire, il n'est même pas nécessaire de posséder un ordinateur :un simple générateur d'ondes suffit. Il est également possible de régler celui-ci de façon à ce que la perte de rendement ne soit pas trop importante et par ce fait, moins évidente (par exemple, 1 trame sur 10 est systématiquement rejetée). Il est évident que pour ce genre d'attaque, il est nécessaire d'avoir un accès physique à une prise du réseau.

[3] Une attaque de rejet (refus) de service est une attaque visant à rendre le service offert par le protocole indisponible pour l'utilisateur.

Ethernet
Conflit d'adresse

Ici, on installe une machine possédant une adresse Ethernet existante. Cette adresse peut être réalisée par divers moyens électroniques et informatiques. Comme pour l'attaque précédente, un accès physique aux câbles du réseau est indispensable.

Ethernet
Host-Spoofing[4]

Ici encore, on installe une machine possédant une adresse Ethernet existante, mais on veille à n'utiliser la machine pirate qu'aux moments ou la machine cible est « down ». Ici, pas de conflit d'adresse ; pour le réseau local, le pirate est devenu crédible.

Ethernet
Eavesdropping[5]

Dans ce cas, il n'est même pas nécessaire d'avoir une machine possédant une adresse Ethernet, il suffit d'écouter et de pouvoir faire la distinction entre les différentes trames et les différents destinataires.

Pour contrer ce genre d'attaque, il convient d'adopter une politique rigoureuse en matière d'accès et de gestion des prises de connexion actives. Ces attaques ne peuvent en effet réussir que si le pirate a accès physiquement au câble Ethernet.

2.2.2 Protocoles de niveau Réseau (Internet).

2.2.2.1 Le protocole IP.

2.2.2.1.1 Rappel.

Les paquets IP sont à la base du fonctionnement de la paire TCP/IP. Chaque paquet contient l'adresse 32_bits de la source et de la destination, des bits du champ d'option, une somme de contrôle de l'entête (header checksum) et des données. Un paquet IP typique a une longueur de quelques centaines de bytes. Ces paquets

[4] C'est essayer de se faire passer pour quelqu'un d'autre.
[5] Ce sont les écoutes indiscrètes.

transitent par milliards à travers le monde, sur des réseaux Ethernet, X25, FDDI Ring, ect.

Rappelons que nous sommes dans le cas des réseaux non-connectés à commutation par paquets. Nous n'avons donc aucune garantie que les paquets envoyés seront délivrés ou, s'ils le sont, qu'ils le seront dans l'ordre. Il n'y a aucune vérification que les paquets ont été transmis sans erreur. La somme de contrôle de l'entête ne concerne en effet, comme son nom l'indique, que l'entête lui-même.

Protocole : IP
HOST-SPOOFING[6]

En théorie, n'importe quel hôte peut transmettre un paquet avec n'importe quelle adresse source. Il n'y a donc aucune garantie que le paquet reçu provient bien de l'adresse source reprise dans le paquet en question.[7] L'authentification et la sécurité en général doivent utiliser des mécanismes qui relèvent de couches de protocoles de plus hauts niveaux.

Un paquet qui voyage sur une longue distance le fera par sauts successifs. Chaque saut le fera aboutir dans un router ou dans un hôte qui redirigera le paquet pour son prochain saut en fonction des informations de routage. Durant le trajet, le paquet peut être fragmenté en morceaux plus petits si le fragment original est trop grand pour le saut à venir.

Aucun réassemblage ne sera effectué entre la source et la destination.
Un router peut laisser tomber des paquets s'il est trop surchargé.
Les paquets peuvent arriver dans n'importe quel ordre.
Les paquets peuvent arriver en plusieurs exemplaires à la destination.

C'est un des protocoles de plus haut niveau (TCP) qui permet d'obtenir un circuit sûr (*reliable*).

2.2.2.1.2 Utilisation du label de sécurité.

Comme nous l'avons vu, un datagramme possède un certain nombre de champs d'option que l'on peut utiliser : label de sécurité, routage strict à partir de la source (*strict source routing*), routage libre à partir de la source (*loose source routing*), etc. Ceux-ci sont rarement employés dans la pratique, exception faite des applications militaires.

Les labels indiquent le niveau de sécurité des processus expéditeurs et destinataires. En fait, on ne devrait pouvoir employer de paquets sur un medium de

[6] Dans ce cas particulier, on parlera d'IP-SPOOFING
[7] Certains systèmes d'exploitation vérifient que ce champ est correcte.

niveau de sécurité inférieur. On ne devrait également ne pas pouvoir lire d'informations en provenance de ligne de niveau de sécurité plus élevé.

En fait, au sein d'un réseau, le but principal recherché par le label de sécurité est de guider les décisions de routage. Un paquet marqué TOP-SECRET ne devrait pas pouvoir transiter par des lignes prévues avec un niveau de sécurité plus bas.

2.2.2.1.3 IPng (IP version 6)

Outre le passage d'un adressage de 32 à 128 bits, avec IPng est fourni un mécanisme de sécurité intégré pour l'authentification et le chiffrement. Ce mécanisme cryptographique permet de vérifier l'adresse d'un expéditeur. Cette évolution d'IP permettre peut-être de résoudre certains problèmes de sécurité.

2.2.2.2 Protocoles de résolution d'adresse.

2.2.2.2.1 Protocole ARP.

Si on considère, par exemple, les réseaux Ethernet, les adresses IP de 32-bits n'auront aucune signification. En effet, sur réseaux Ethernet, les adresses admises ont une longueur de 48 bits. Il doit donc exister quelque part un driver IP qui va convertir les adresses IP destinataires en adresses Ethernet destinataire. Cela peut se faire statiquement ou à l'aide d'algorithmes mais généralement en consultant des tables. Le protocole chargé de faire la correspondance est le protocole ARP (*Address Resolution Protocol*).

ARP fonctionne en diffusant à travers tout le réseau Ethernet (ou autre), un paquet Ethernet contenant l'adresse IP en question. Le destinataire (unique) répond à cette requête en renvoyant un paquet contenant son adresse IP et son adresse Ethernet.

Protocole : ARP
Host-Spoofing

Il peut régner une certaine insécurité sur ce réseau si des noeuds non certifiés ont des droits d'écriture sur le réseau local. Une telle machine pourrait émettre des messages ARP et diriger tout le trafic vers elle. Cette machine pourrait alors se faire passer pour une autre ou modifier les donnés qui la traversent.

Protocole : ARP
Variante d'HOST-SPOOFING

Si le réseau local utilise le protocole de résolution d'adresse ARP, des formes plus subtiles d'host-spoofing sont possibles. En particulier, il devient trivial d'intercepter, de modifier et de faire suivre des

paquets. Il est tout aussi facile de prendre possession du rôle de l'hôte ou d'espionner tout le trafic du réseau.

Protocole : ARP
Rejet de service

Il est possible de lancer des attaques de refus de service en déclenchant des broadcast storm. Il existe plusieurs façons de faire cela ; c'est assez facile à réaliser si la plupart ou tous les hôtes sur le réseau jouent le rôle de passerelle. Un pirate peut émettre un paquet à l'intention d'un hôte qui n'existe pas. Chaque hôte, dès qu'il reçoit ce message, va tenter de le faire suivre jusqu'à sa destination. Ce simple fait en lui-même constitue une quantité significative de trafic car chaque hôte va générer une requête ARP pour résoudre l'adresse de destination. Le pirate peut poursuivre en diffusant (broadcast) une réponse ARP proclamant que l'adresse pour atteindre cette destination est l'adresse de diffusion (broadcast)sur tout l'Ethernet. Chaque hôte ne recevra donc pas seulement le paquet falsifié mais également de nombreuses copies de celui-ci originaires de tous les autres hôtes du réseau.

Le mécanisme ARP est généralement automatique. Sur les réseaux spéciaux de sécurité, la correspondance entre les adresses IP et les adresses Ethernet peut se faire sur un câblage indépendant et le protocole automatique ARP est alors supprimer pour prévenir toute interférence.

2.2.2.2.2 Protocole RARP.

Un système en train de *booter* est une cible tentante ; si un pirate arrive à corrompre le processus de *bootage*, un nouveau noyau possédant des mécanismes de protection altérés peut-être substitué.

Le bootage basé sur RARP est risqué car il repose sur des réseaux du type Ethernet, avec toutes les vulnérabilités que cela implique. On peut apporter une petite amélioration de sécurité en s'assurant que la machine qui *boot* utilise un numéro de port source UDP aléatoire ; sinon, un pirate pourrait se faire passer pour un serveur et envoyer des paquets de données falsifiées.

La plus grande mesure de protection que l'on puisse avoir est en fait inhérente au principe du *bootage*. En effet, le pirate n'aura généralement qu'une seule chance ; un système en train de *booter* ne reste pas longtemps dans cet état. Cependant, si les communications entre clients et serveurs peuvent être interrompues, une plus grande liberté d'attaque sera laissée au pirate.

2.2.2.3 Protocole ICMP

ICMP est l'outil de base pour l'administration des réseaux fonctionnant avec TCP/IP. C'est un mécanisme de bas niveau utilisé pour influencer le comportement

des connexions TCP et UDP. Il peut être utilisé pour informer les hôtes d'une meilleure route pour atteindre une destination donnée, pour indiquer des problèmes rencontrés avec une route particulière ou pour clôturer une route à cause d'un problème de réseau[8].

Protocole : ICMP
Points Faibles

Un bon nombre de messages ICMP reçus par un hôte sont spécifiques à une connexion particulière ou sont déclenchés par un paquet envoyé par cet hôte. Dans de tels cas, on inclut dans le message ICMP l'entête du datagramme IP et les 64 premiers bits de l'entête de transport. Il faut limiter la portée des changements dictés par ICMP. En effet, un message de REDIRECT ou un message de UNREACHABLE DESTINATION devraient être spécifiques à chaque connexion. Malheureusement, beaucoup d'implémentations d'ICMP n'utilisent pas cet information supplémentaire. Quand de tels messages arrivent, toutes les connexions entre les deux hôtes sont affectées. Quand on reçoit un paquet de UNREACHABLE DESTINATION disant qu'un certain paquet ne peut atteindre l'hôte x.com, toutes les connexions vers x.com seront détruites (voir attaque de rejet de service ci-dessous). Ceci est vrai, même si ce message a été provoqué par un filtre spécifique au port d'un firewall(cfr. firewalls). Il serait alors commode que le firewall s'empêche de générer des messages ICMP qui pourraient détruire des appels légitimes provenant d'une machine. Il faut noter qu'une partie de la communauté des pirates est adepte de l'abus de message ICMP à des fins de destruction de connexions.

La plupart des attaques ICMP sont faciles à esquiver avec un minimum de paranoïa. Si un hôte prend soin de vérifier qu'un message porte bien sur une connexion particulière, la plupart de ces attaques échoueront. Dans le cas de TCP, cela inclus la vérification que le paquet ICMP contient un numéro de séquence plausible dans la portion du paquet qui est retournée. Cependant, ces vérifications ne sont pas applicables à UDP.

[8] ICMP supporte également l'outil de monotoring, pour administrateurs de systèmes ou de réseaux, le plus important : le programme PING.

Protocole : ICMP
Attaque REDIRECT.

On peut encore faire pire avec les messages REDIRECT 9: tout qui peut altérer vos connaissances de la route la plus appropriée pour atteindre une destination peut probablement pénétrer votre machine.

La complication vient du fait qu'un message REDIRECT doit être attaché à une connexion existante ; il ne peut pas être utilisé pour effectuer des changements des tables de routage qui n'auraient pas de rapport avec les connexions en cours. De plus, les messages REDIRECT sont seulement applicables dans une topologie limitée ; ils ne peuvent être envoyés que de la première passerelle sur le chemin vers l'hôte d'origine. Une passerelle plus éloignée ne peut ni informer cet hôte, ni utiliser les messages ICMP REDIRECT pour contrôler d'autres passerelles.

Supposons qu'un intrus ait pénétré une passerelle secondaire d'un hôte particulier, mais pas le primaire10. Supposons également que l'intrus désire établir une fausse route jusqu'à l'hôte de confiance (T) au travers de cette passerelle secondaire. Il faut suivre la séquence d'opérations suivante :

- Envoyer un faux paquet d'ouverture TCP à la cible en le proclamant originaire de T.
- La cible va répondre en envoyant son propre paquet d'ouverture en le routant à travers la passerelle primaire, qui elle est sûre.
- Alors que ce message est en route, un faux message REDIRECT peut être envoyé, se revendiquant en provenance de la passerelle primaire et se référant à la fausse connexion.
- Ce paquet apparaîtra comme étant un message légitime de contrôle et donc les changement de routage qu'il contient seront acceptés.
- Si l'hôte cible effectue ces changements dans ses tables de routages globales en plus de sa mémoire cache, l'intrus pourra opérer en se faisant passer pour l'hôte T.

[9] Ces messages sont utilisés par les passerelles pour indiquer aux hôtes de meilleures routes. Comme tel, ICMP peut également être abusé de la même manière que RIP peut l'être.

[10] Il peut suffir de pénétrer un hôte ordinaire sur le réseau local de la cible et de le revendiquer comme étant un gateway).

Certains hôtes ne vérifient pas suffisamment les messages ICMP REDIRECT ; dans de tels cas, l'impact de ces attaques devient semblable à celles basées sur RIP.

Se défendre contre les attaques REDIRECT mérite une attention toute particulière car elles peuvent être beaucoup plus dommageables. La meilleure solution est probablement de restreindre les changements de route à la connexion spécifiée ; la table de routage générale ne devrait également ne pas pouvoir être modifiée en réponse à des paquets ICMP REDIRECT.

Finalement il est bon de se poser la question de savoir si les messages ICMP REDIRCT sont utiles dans les environnements d'aujourd'hui. En effet, ils sont seulement utilisables sur des réseaux locaux ayant plus d'une passerelle entre eux et l'extérieur. Cependant, il est facile de maintenir une information de routage locale correcte et complète.

Protocole : ICMP
Refus de service

ICMP peut également être utilisé pour l'attaque de rejet de service (denial of service attack). Certains de ces messages, comme DESTINATION UNREACHABLE et TIME TO LIVE EXCEEDED, peuvent être utilisés pour réinitialiser (reset) une connexion existante. Si l'intrus connaît les numéros de port de la connexion TCP de la machine locale et de la machine distante, un paquet ICMP dirigé vers une de ces connexion peut être falsifié.

Une attaque de rejet de service plus générale peut être lancée en envoyant un message SUBNET MASK REPLY frauduleux. Certains hôtes accepteront ce message, qu'ils aient envoyé une requête ou non ; un faux message pourrait bloquer toutes les communications avec l'hôte cible.

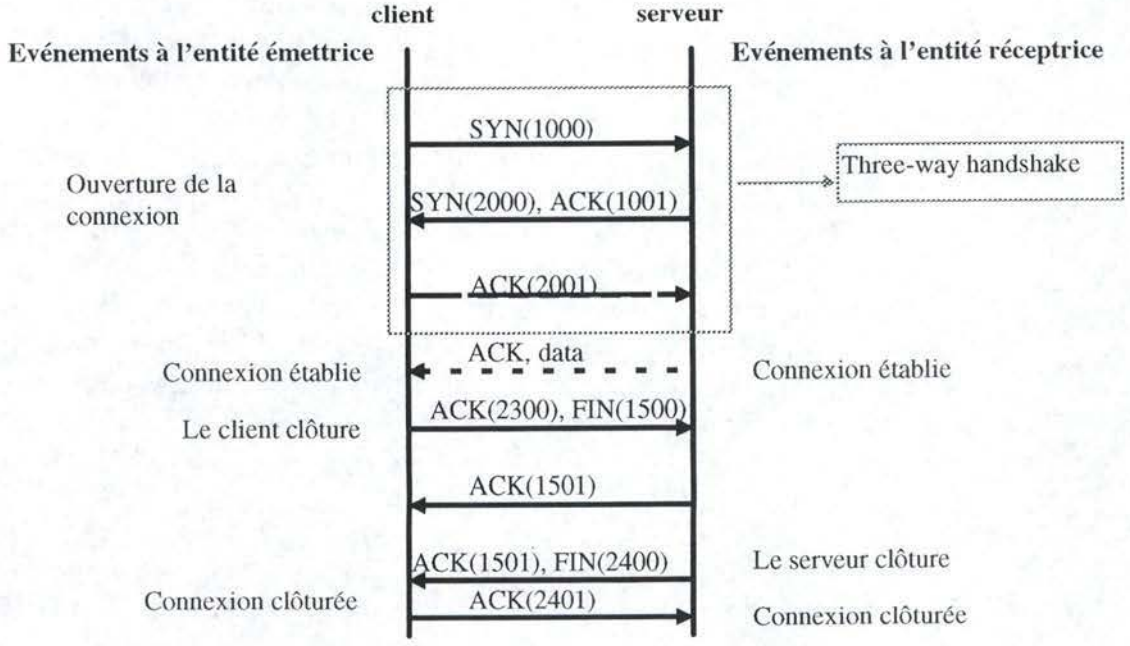
Les attaques SUBNET MASK peuvent être bloquées si le paquet réponse est honoré seulement au moment approprié. En général, un hôte ne désire voir ce genre de message qu'au moment du *boot* et seulement s'il suit une requête de celui-ci ; une vieille réponse ou une réponse qui ne fait pas suite à une requête ne devrait jamais être acceptée. Il y a peu de défense contre un réponse falsifiée répondant à une authentique requête SUBNET MASK. Un hôte qui a envoyé une telle requête a peu de ressources à sa disposition qui lui permettent de vérifier la réponse. Cependant, si l'authentique réponse n'est pas bloquée par l'intrus, la cible va recevoir des réponses multiples ; une vérification que toutes ces réponses sont acceptables nous préservera contre cette attaque (et également contre les erreurs d'administration).

2.2.3 Protocoles de niveau Transport.

2.2.3.1 Protocole TCP.

Le protocole de transport (TCP) fournit des circuits virtuels sûrs (*reliable*) aux processus utilisateurs. Les paquets perdus ou endommagés sont retransmis; les paquets entrant sont réordonnés si nécessaire.

L'ordre est maintenu grâce au numéro de séquence (*sequence number*) contenu dans le paquet. Chaque *byte* de données envoyé, tout comme chaque paquet d'ouverture ou de fermeture de communication, est numéroté individuellement.



Cette figure représente un exemple d'une session TCP. Le paquet initial, avec le bit SYN (SYNchronize ou requête d'ouverture) activé, transmet le numéro de séquence initiale pour son côté de la connexion. Le numéro de séquence initiale est aléatoire. Tous les paquets suivants auront le bit ACK (ACKnowledge, ou accusé de réception) activé.

Tous les paquets envoyés, exception faite du tout premier, contiennent un numéro d'accusé de réception *(acknoledgment)*; c'est le numéro de séquence des derniers *bytes* reçus.

Chaque paquet TCP contient, entre autres, quatre champs particuliers :
< adresse de **l'hôte local** ; numéro de port de **l'hôte local** ;
adresse de l'hôte **destinataire** ; numéro de port **destinataire** >

Ce quadruplet, appelé *socket*, identifie un circuit virtuel particulier. Il n'est donc pas interdit d'avoir plusieurs circuits correspondant à un même port sur une même machine. Il suffit que l'adresse du destinataire ou que le port du destinataire diffèrent.

Les serveurs (processus qui désire fournir des services via TCP) écoutent sur des ports particuliers. Par convention, les numéros de ces ports sont petits. Cette convention n'est pas toujours respectée, d'où certains problèmes de sécurité qui seront abordés plus loin. Les numéros de port pour tous les services appelant sont supposés connu par le processus appelant. Un port à l'écoute est en quelque sorte à moitié ouvert; seuls l'adresse locale (ou une des adresses locales) et les numéros de ports sont connus. Ce n'est que quand un paquet de requête de connexion arrivera que les champs manquant seront enfin complétés. Si la connexion est autorisée, le système d'exploitation va cloner la connexion d'écoute pour que d'autres requêtes puissent être satisfaites.

Les clients utilisent, eux, les services offerts par les serveurs. Un client demande rarement à utiliser un port particulier, bien qu'il puisse le faire. Il reçoit généralement la réponse sur un port qui lui a été désigné par le système d'exploitation.

Les protocoles TCP et UDP de Berkeley possèdent la notion de « ports privilégiés ». Ce sont les ports dont les numéros sont inférieurs à 1024 et qui ne peuvent être attribués qu'à des processus privilégiés[11]. Cette restriction fait partie du mécanisme d'authentification. Le but recherché par l'emploi de ces ports privilégiés est qu'un système distant (*remote*) puisse avoir confiance en l'authenticité des informations écrites sur de tels ports. Cependant, ni les spécifications de TCP, ni celles d'UDP ne contiennent un tel concept. Ce concept n'a d'ailleurs aucune signification pour un ordinateur individuel. Les administrateurs ne devraient jamais se fier aux schémas d'authentification de Berkeley quand ils dialoguent avec de telles machines.

Les numéros de séquence mentionnés plus haut ont une autre fonction. Parce que le numéro de séquence change constamment pour chaque nouvelle connexion, il est possible pour TCP de détecter de vieux paquets d'occurrences précédentes du même circuit (p.e. : utilisation précédante du même quadruplet identifiant). Une connexion ne peut donc être complètement établie tant que chacune des deux parties n'a pas envoyé un accusé de réception du numéro de séquence initial à l'autre.

Protocole : TCP
Attaque SEQUENCE NUMBER.

Si un pirate peut prédire la séquence initiale de sa cible, alors il est possible, pour l'agresseur, de faire croire à la cible qu'elle communique avec une machine de confiance (trusted). Dans ce cas, les protocoles qui reposent sur l'adresse IP pour l'authentification (p.e. : Rlogin) peuvent être exploités pour pénétrer le système cible. Cette attaque est connue sous le nom de Sequence Number Attack[12].

[11] La plupart des versions TCP et UDP pour les systèmes UNIX renforce la règle que seul le super-utilisateur (*root*) peut créer des numéros de port inférieurs à 1024.
[12] Morris

La séquence normale d'établissement d'une connexion TCP comprend une Three-way Handshake (triple poignée de mains). Le client (C) choisit et transmet son numéro de séquence initial ISNc, le serveur (W) accuse réception et envoie son propre numéro de séquence ISNs. Le client, à son tour, accuse réception de ce paquet. Après cela la connexion est établie et les données peuvent être envoyées.

C → S : SYN(ISNc)
S → C : SYN(ISNs), ACK(ISNc)
C → S : ACK(ISNs)
C → S : données et/ou
S → C : données

Ceci signifie que C doit d'abord avoir reçu ISNs avant que la conversation puisse débuter.

Supposons qu'il soit possible pour un intrus X de prédire ISNs. Dans ce cas, X pourrait envoyer la séquence suivante afin de se faire passer pour un hôte de confiance T (trusted) :

X → S : SYN(ISNx), SRC=T
S → T : SYN(ISNs), ACK(ISNx)
X → S : ACK(ISNs), SRC=T
X → S : ACK(ISNs), SRC=T, mauvaises données

Bien que le message S→T n'arrive jamais à X, X est cependant capable de deviner son contenu et peut ainsi envoyer des données. Si X pratique cette attaque sur une connexion qui permet l'exécution de commandes (comme le serveur rsh de Berkeley), il est alors possible d'exécuter des commandes malicieuses.

Le message S→T : SYN(ISNs), ACK(ISNx) ne disparaît pas comme par enchantement ; en fait, le véritable T recevra ce message et essaiera de réinitialiser la connexion. Il suffit de submergé le port serveur de T avec des demandes de connexions. Cela va générer un overflow qui va faire en sorte que le message S→T va être perdu. On peut également attendre que la machine T soit déconnectée ou rebootée.

<u>Question</u> : Comment peut prédire l'ISN qui est, rappelons-le, aléatoire ?

Réponse : ***Dans les systèmes Berkeley, le numéro de séquence initial est incrémenté qu'une même constante une fois par seconde, et par la moitié de cette constante à chaque ouverture de connexion. Donc, si on observe l'ISNs utilisé lors d'une connexion légitime, on pourra calculer, avec un grand degré de sûreté, l'ISNs' qui sera utilisé pour la prochaine tentative de connexion.***

Des solutions simples sont l'augmentation de la fréquence d'incrémentation, l'emploi d'un incrément aléatoire ou l'emploi d'algorithmes de cryptographie pour la génération d'ISNs (p.e. : *DES, Data Encryption Standard*).

2.2.3.2 Protocole UDP.

UDP offre aux programmes d'application le même niveau de service que celui utilisé par l'IP. La livraison du paquet est basée sur le principe de « faire de son mieux » (*best-effort*) ; il n'y a pas de correction d'erreur, pas de retransmission, pas de détection de paquets perdus, dupliqués ou en désordre. En fait, la détection d'erreur est optionnelle en UDP.

Pour compenser ces désavantages, les services de niveaux supérieurs sont allégés. En particulier, il n'y a pas d'établissement d'une connexion. Ceci rend UDP particulièrement adapté aux applications requête/réponse[13], dans lesquelles le nombre de messages échangés est petit par rapport à celui engendré par le mode d'établissement de liaison de TCP.

Cependant, quand UDP est utilisé pour de grandes transmissions, il tend à mal se comporter sur un réseau. Ce protocole manque en effet de possibilité de contrôle de flux. Il peut donc rapidement submerger un hôte ou un routeur et causer ainsi la perte de nombreux paquets.

UDP utilise les mêmes conventions de serveur et de numéros de port que TCP, mais dans des espaces d'adresses différents. De la même manière, les serveurs utilisent d'habitude les ports de petits numéros (mais pas obligatoirement). Il n'y a pas de notion de circuit . Tous les paquets destinés a un port de numéro donné sont envoyés au même processus, sans tenir compte de l'adresse ou du numéro de port.

Protocole : UDP
Comparaison avec TCP
Il est beaucoup plus facile d'imiter des paquets UDP que des paquets TCP car il n'y a ni handshake, ni numéro de séquence.

[13]Par exemple : la transmission de séquences d'images à distance où il n'est pas grave d'avoir de petites erreurs de temps en temps dans l'image.

Il faut donc prendre un soin tout particulier quand on utilise des adresses sources issues de ces paquets. Les applications doivent elles-mêmes assurer leurs propres authentifications.

2.2.4 Protocoles de niveau Application.

2.2.4.1 FTP

2.2.4.1.1 Introduction

FTP en lui même ne contient pas de défaut de protection. Cependant, certains aspects de l'implémentation méritent un soin particulier.

FTP sert de support à la transmission et à la traduction de textes et fichiers binaires en ensembles de caractères. Dans une session typique (figure ci-dessous), la commande ftp de l'utilisateur ouvre un canal de contrôle jusqu'à la machine cible. Les lignes commençant par → indique les commandes qui sont envoyées à ce moment sur le fil ; les réponses sont précédées d'un code de trois chiffres.

```
$ ftp -d research.att.com
220 inet FTP server (Version 4.271 Fri Apr 9 10 :11 :04 EDT 1993) ready.
→USER anonymous
331 Guest login ok, sent ident as password.
→PASS guest
230 guest login ok, access restrictions apply.
→SYST
215 UNIX Type : L8 Version :BSD-43
Remote system type is UNIX.
→TYPE I
200 Type set to I
Using binary mode to transfer files.
Ftp> ls
→PORT 192,20,225,3,5,163
200 PORT command successful.
→TYPE A
200 Type set to A.
→NLST
150 Opening ASCII mode data connection for /bin/ls.
Bin
dist
etc
ls-lR.z
netlib
pub
226 Transfer complete
→TYPE I
200 Type set to I.
ftp>bye
→QUIT
221 Goodbye.
$
```

Exemple de session FTP.

Parfois, comme après que la commande USER ait été envoyée, le code réponse indique que le démon est entré dans un état spécial où seulement certaines commandes sont acceptées.

La donnée en cours, qu'il s'agisse d'un transfert de fichier ou de la commande de listing d'un répertoire, est envoyée sur un canal de données différent du canal de

contrôle. C'est le port numéro 20 que le serveur utilise à cette fin[14]. Par défaut, le client utilise le même numéro de port que celui utilisé pour le canal de contrôle. La spécification du protocole FTP suggère qu'un seul canal soit créé et maintenu ouvert pour tous les transferts de données pendant la session. La plupart des implémentations créent une nouvelle connexion pour chaque fichier. De plus, pour l'une des raisons les plus obscures de TCP (état TIMEWAIT), un numéro de port différent doit être utilisé à chaque fois. Habituellement, le client écoute sur un numéro de port au hasard et informe le serveur de cela via la commande PORT. A son tour, le serveur appelle le port en question.

2.2.4.1.2 L'authentification FTP.

FTP repose sur l'introduction d'un *login* et d'un mot de passe pour l'authentification. Les simples mots de passe semblent de plus en plus inadéquats ; de plus en plus de sites adoptent le principe du mot de passe utilisable une seule fois. Rien n'empêche l'emploi d'une méthode d'authentification dans FTP. Il est vital, cependant, que la réponse 331 à une commande USER soit affichée à l'écran ; on présume que ce message devrait contenir le challenge(voir chapitre authentification). Une implémentation FTP qui maintient secrète cette réponse ne devrait pas être utilisée de cette façon ; si de telles implémentations deviennent courantes, il peut devenir nécessaire d'utiliser un nouveau code réponse pour indiquer que l'utilisateur doit voir le contenu du challenge.

2.2.4.1.3 FTP anonyme (Anonymous FTP).

Une deuxième zone problématique est le FTP anonyme. Le FTP anonyme, bien que non obligatoire dans une spécification FTP, fait partie des trésors de la tradition d'Internet. Le FTP anonyme est en effet devenu un des principaux standards sur l'Internet pour la publication de software, d'articles, d'images, etc. La plupart des grands sites ont besoin d'avoir un « dépôt » anonyme FTP accessible au public. C'est parfois une nécessité et non un résultat de leur seule volonté.

Il n'y a pas de doute que le FTP anonyme est un service de valeur. C'est indiscutablement, après le courrier électronique, le service le plus important d'Internet. Il faut cependant veiller à l'administrer avec beaucoup de soin.

Une partie du problème provient de la technique d'implémentation choisie. Certaines implémentations de FTP nécessitent la création d'une réplique partielle de l'arbre des répertoires ; on doit prendre soin à ce que ces fichiers ne soient pas corruptibles. Ils ne doivent pas contenir non plus d'informations sensibles comme des mots de passe cryptés.

Un second problème est que le FTP anonyme est véritablement anonyme ; il n'y a aucun enregistrement du « qui fait quoi ». Les serveurs de courrier (mail servers) fournissent, eux , de telles données ; ils fournissent également des techniques utiles pour la limitation de la charge, etc.

[14] Cfr. Tableau listant tous les numéros de ports utilisés par TCP et UDP

Protocole : FTP
Refus de service.

Il est possible de rendre un hôte temporairement inutilisable par l'envoi massive de requêtes FTP. Si cet acte est délibéré, il peut être considéré comme une attaque de refus de service tout à fait valable.

Avec FTP, la première règle, et sans doute la plus importante, est qu'aucun fichier ou répertoire couvert par la zone du FTP anonyme ne peut être ni accessible en écriture, ni possédé par le *login* FTP parce que le FTP anonyme s'exécute avec cet identifiant d'utilisateur. Considérons l'attaque suivante :

Protocole : FTP
Changement de droits

Ecrivons un fichier s'appelant .rhosts sur le répertoire home de FTP. Utilisons ensuite ce fichier pour autoriser une connexion RSH comme FTP sur la machine cible. Si le répertoire FTP n'est pas accessible en écriture mais appartient à FTP, il faut être prudent : certains serveurs permettent aux hôtes distants de changer les droits d'accès.

Il est fortement conseillé de supprimer du protocole FTP *anonymous* les lignes de code correspondant à cette possibilité.

La règle suivante est d'éviter de laisser un vrai fichier /ect/passwd dans la zone couverte par le FTP anonyme. Sinon :

Attaque semblable à celle du TFTP expliquée plus loin.

Créer ou non un répertoire public accessible en écriture est une décision difficile à prendre. Bien que cette disposition soit, à n'en pas douter, d'une très grande utilité, de nombreux pirates ont trouvé des moyens d'en abuser.

Protocole : FTP
Entrepôt pour pirates

On peut ainsi découvrir que sa propre machine est devenue un entrepôt pour les software et les images pornographiques des pirates. Cet entrepôt peut être permanent ou momentané. Dans ce dernier cas,

l'emploi de cet entrepôt se fait dans un but d'anonymat : une personne enregistre des données dans ce répertoire, et informe une autre personne que ces données sont bien arrivées, l'autre personne peut alors venir les chercher et effacer toute trace de leur passage.

2.2.4.2 TFTP : Trivial File Transfert Protocol.

TFTP est un simple mécanisme de transfert de fichier basé sur UDP. Il permet le transfert de fichiers sans authentification. Il est utiliser pour *booter* des stations de travail sans disque dur et les terminaux X11. N'importe quel fichier accessible en lecture publique peut être lu du monde entier ; c'est à l'implémenteur et/ou à l'administrateur du système qu'il incombe la responsabilité de rendre cet univers aussi petit que possible. Un démon TFTP correctement configuré limite les transferts de fichiers à un ou deux répertoires, typiquement /usr/local/boot et X11 font library.

Protocole : TFTP
Copie du fichier password

Il y a quelques années, la plupart des fabriquants fournissaient leur software TFTP avec des accès illimités, ce qui rendait la tâche des pirates très facile :

```
$ tftp target.cs.boofhead.edu
tftp> get /etc/passwd/tmp/passwd
Received 1205 bytes in 0.5 seconds
$ crack </tmp/passwd
```

Ce serait trop facile. En effet, un dictionnaire de mot de passe typique appliqué à ce fichier donnerait environ 25% des mots de passe. Cela rendrait la machine considérée, et celles pour lesquelles elle est une machine de confiance (trusted), complètement vulnérables. Il vaut mieux éviter de faire tourner TFTP sur sa machine à moins que cela soit véritablement indispensable. Dans ce cas, il faut s 'assurer que TFTP est configuré correctement afin de ne délivrer que le bon fichier et au bon client.

Protocole : TFTP
Configuration routeur

Certains routeurs utilisent TFTP pour charger des images exécutables ou des fichiers de configuration. Ce dernier cas est particulièrement risqué, pas seulement parce qu'un pirate sophistiqué pourrait générer un faux fichier (en général, ce sera très difficile) mais surtout par ce que

les fichiers de configuration contiennent souvent des mots de passe. Un démon TFTP utilisé pour remplacer de tels fichiers devra être installé de telle sorte que seul le routeur puisse lui parler.

2.2.5 Routage et protocole de routage

Les protocoles de routage sont des mécanismes de recherche d'un chemin approprié à travers Internet. Ils sont essentiels au bon déroulement de TCP/IP. L'information de routage établit deux chemins : de la machine appelante vers la machine appelée et le chemin de retour. En fait, ce sont généralement les mêmes[15]. Du point de vue de la sécurité, c'est le chemin de retour qui est le plus important. Si le pirate peut altérer le mécanisme de routage, on peut alors faire croire à la cible que la machine du pirate est une machine de confiance (*trusted*), les flux d'informations revenant en sens inverse étant redirigés vers celui-ci. Si cela arrive, le mécanisme d'authentification qui repose sur la vérification de l'adresse source échoue.

L'abus des mécanismes et des protocoles de routage est probablement la plus simple des attaques disponibles. Il existe de nombreuses façons d'y arriver, cela dépendant des protocoles de routage utilisés. Quelques unes de ces attaques réussissent seulement si l'hôte distant pratique une authentification basée sur l'adresse ; d'autres attaques employées sont plus puissantes.

Les attaques décrites ci-dessous peuvent également servir pour pratiquer des attaques de refus de service (*denial of service attack*) en modifiant les tables de routage d'un hôte ou d'une passerelle.

Protocole de routage
Attaque IP SOURCE-ROUTING.

Le mécanisme d'attaque le plus simple à utiliser pour le piratage est le routage à partir de la source IP (IP SOURCE ROUTING).

Il y a de nombreuses possibilités d'attaques des mécanismes de routage standards. La plus simple utilise l'option IP LOOSE SOURCE ROUTE. C'est en effet à l'aide de cette option qu'une personne à l'origine de l'ouverture d'une connexion TCP peut imposer le chemin que doivent suivre les paquets jusqu'à leur destination16 (court-circuitant ainsi le

[15] S'ils sont différents, on parle de routage asymétrique.
[16] D'après la RFC 1122, la machine destinataire doit utiliser le même chemin pour le retour, que cela ait un sens ou non.

processus de calcul d'itinéraire). Ce qui signifie qu'un pirate peut choisir n'importe quelle adresse IP source, y compris celle d'une machine de confiance (trusted) sur le réseau local de la cible. Toutes les ressources, matérielles ou logicielles, de ces machines deviennent alors également disponibles pour le pirate.

Le moyen de défense le plus facile est de rejeter les paquets contenant l'option de routage à partir de la source. Beaucoup de routeurs offrent cette possibilité de protection. Le routage a partir de la source est rarement employé à bon escient mais le debuggage de certains problèmes réseau peut nécessiter le recours à cette technique. Cependant, cela vaut peut-être la peine de sacrifier cette option pour des raisons de sécurité. Certaines versions de Rlogind et Rshd sont prévues pour rejeter des connexions avec routage à partir de la source.

Un variante de cette solution serait d'analyser la route indiquée par la source et de l'accepter seulement si toutes les passerelles présentes sur le chemin sont des passerelles de confiance *(trusted)* ; de cette manière, la dernière passerelle serait certaine de délivrer le paquet à l'authentique hôte de destination. Cependant, la complexité de cette solution n'en vaut peut-être pas la peine.

Certains protocoles (p.e.: rlogin et rsh de Berkeley) permettent aux utilisateurs ordinaires d'étendre leur confiance à des combinaisons hôte/utilisateur distantes. Dans ce cas, les utilisateurs individuels, plutôt que le système tout entier, peuvent être victimes d'attaques SOURCE-ROUTING.

A part cette solution, il est assez difficile de se défendre contre ce genre d'attaque. La meilleure idée serait que les passerelles internes au réseau local rejettent les paquets en provenance de l'extérieur et qui se disent en provenance du réseau local lui-même[17]. Ceci est en fait moins pratique qu'il ne semble depuis que certains adaptateurs de réseaux Ethernet reçoivent leur propres transmissions (cette caractéristique est du ressort d'un protocole de plus haut niveau). De plus, cette solution échoue complètement si une organisation possède deux réseaux de confiance connectés via un pivot multi-organisations. On ne peut peut-être pas avoir autant confiance en d'autres utilisateurs sur ce pivot qu'en les utilisateurs du réseau local. Ces autres utilisateurs sont peut-être également plus vulnérables aux attaques externes. Il vaut mieux cependant éviter ce type de topologie.

[17] C'est la solution que BELNET a adoptée. BELNET assure de plus au réseau des facultés que tout message provenant de l'extérieur et possédant une adresse source et une adresse de destination appartenant au réseau des facultés sera rejeter. Rappelons que le réseau des facultés n'est connecté au monde extérieur que via BELNET.

Protocole : R.I.P.
HOST-SPOOFING

Le protocole d'information de routage (RIP) est employé pour propager les informations de routage sur les réseaux locaux et plus spécialement sur les media de diffusion. Typiquement, l'information reçue n'est pas vérifiée. Ceci permet à un intrus d'envoyer de fausses informations vers un hôte cible et à chacune des passerelles le long du chemin, lui permettant ainsi de pouvoir se faire passer pour un hôte particulier. La plus commune des ces attaques consiste à employer un chemin vers un hôte inutilisé plutôt que vers un réseau ; ceci devrait permettre à tous les paquets destinés à cet hôte d'aboutir dans la machine du pirate. En effet, tenter de dérouter tous les paquets d'un réseau tout entier serait trop visible ; en comparaison, essayer de se faire passer pour une station de travail tournant au ralenti est beaucoup moins risqué. Une fois que cela est fait, tous les protocoles qui fonctionnent sur base des adresses pour l'authentification seront compromis.

Cependant, cette attaque peut être rendue plus subtile et plus sérieuse, ce qui est tout bénéfice pour le pirate. Supposons, dans ce cas, que le pirate emploie plutôt un chemin vers un hôte ou une station de travail toujours active. Tous les paquets destinés à cette machine seront routés vers la machine du pirate pour inspection et possible altération. Ils seront ensuite ré-émis en utilisant le routage par adresse IP de la source vers la destination originelle (source routing). Un pirate peut dès lors capter les mots de passe et autres données sensibles. Ce mode d'attaque est unique dans la mesure où il affecte également les appels vers extérieur ; donc un utilisateur appelant de la machine cible peut, sans le vouloir, divulguer son mot de passe. La plupart des attaques discutées jusqu'ici étaient utilisées pour falsifier un adresse source ; celle-ci est centrée sur l'adresse de destination.

Il est un peu plus facile de se défendre contre une attaque RIP que contre une attaque SOURCE-ROUTING, bien que quelques unes de ces défenses soient similaires. Une passerelle paranoïaque (qui filtre les paquets en se basant sur les adresses de source ou de destination[18]), bloquera toutes les tentative d'host-spoofing (en ce compris l'attaque SEQUENCE NUMBER de TCP), de telle sorte que les paquets de l'agresseur ne pourront la franchir. Il existe cependant d'autres manières de se défendre contre les attaques RIP.

[18] C'est une technique de ce genre que nous allons développer tout au long de ce travail.

Une des ces méthodes serait, par exemple, d'être un peu plus sceptique sur les routes qu'il accepte. En effet, dans la plupart des environnements, il n'y a aucune raison d'accepter de nouvelles routes vers son propre réseau local[19]. Un routeur qui ferait ces vérifications pourrait facilement détecter ces tentatives d'intrusions. Malheureusement, certaines implémentations reposent sur l'écoute de leurs propres émissions pour maintenir leur connaissance des réseaux qui leur sont directement attachés.

Quelques protocoles de routage, comme RIP version 2 et OSPF (Open Shortest Path First) fournissent un champ d'authentification. Ceux-ci sont cependant d'une utilité limitée pour trois raisons : 1°) le mécanisme d'authentification est un mot de passe. Si le pirate peut se jouer des protocoles de routage, il sera sûrement capable de capter des mots de passe de passage sur le câble Ethernet local.

2°) si un émetteur légitime a été corrompu, alors ses messages, bien qu'authentiques, ne seront pas considérés comme des messages de confiance.

3°) dans la plupart des protocoles de routage, chaque machine ne parle qu'avec ses voisines, celles-ci le répétant aux leurs et ainsi de suite. La tromperie peut donc se propager.

Même si les routeurs locaux n'implémentent pas de mécanismes de défense, les attaques RIP comportent un autre risque : les fausses entrées de routage sont visibles sur une large zone. Tous les routeurs (par opposition aux hôtes) qui recevront les données corrompues vont les retransmettre ; un administrateur d'une collection locale de réseaux, quelque peu suspicieux, pourrait constater l'anomalie. La génération de bons logs[20] peut aider, mais il est difficile de distinguer une véritable intrusion d'une instabilité de routage qui peut accompagner le crash d'une passerelle.

Tous les protocoles de routage ne souffrent pas de tels défauts. Ceux qui impliquent des dialogues entre paires d'hôtes sont plus difficiles à corrompre. Cependant, l'attaque SEQUENCE NUMBER fonctionnera encore.

Une meilleure défense est une défense de type topologique : les routeurs doivent être configurés de telle façon qu'ils conservent en mémoire les routes qui peuvent légitimement apparaître sur un fil donné.

2.2.6 DNS (Domaine Name System).

Le DNS est une base donnée distribuée utilisée pour convertir les noms d'hôtes en adresses IP et inversement. Lors de son déroulement habituel, un hôte envoie une requête UDP au serveur DNS. Ces serveurs répondent en renvoyant soit la réponse attendue, soit de l'information sur des serveurs plus appropriés. Ces requêtes peuvent également être faites via TCP. Celui-ci est habituellement réservé aux transferts de zone (zone transfert). Ceux-ci sont utilisés par les serveurs de back-up pour obtenir

[19] Par exemple, en ce qui concerne les facultés, il n'y a en effet aucune raison qu'un paquet en provenance et destiné aux facultés se balade en dehors du réseau local des facultés.

[20] Enregistrements d'événements particuliers se déroulant sur un réseau. La plupart des tentatives d'intrusions de réseaux font l'objet de tels enregistrements.

une copie complète de leur portion de l'espace des noms. Ils sont aussi utilisés par les pirates pour obtenir rapidement une liste de machines cibles potentielles.

Type	*Fonction*
A	*Adresse d'un hôte particulier*
NS	*Name Server. Délègue un sous arbre à un autre serveur.*
SOA	*Start Of Authority. Désigne le début d'un sous arbre; contient les paramètres de cache et de configuration, et donne l'adresse de la personne responsable de la zone.*
MX	*Mail eXchange. Nomme l'hôte en charge du mail en entrée pour une cible particulière. La cible pourrait posséder une white card de telle sorte qu'un simple enregistrement MX puisse rediriger le mail pour un sous arbre entier.*
HINFO	*Host Type and Operating System Information.*
CNAME	*Un alias pour le nom réel de l'hôte.*
PTR	*Utilisé pour établir la correspondance entre les adresses IP et les noms d'hôtes*

***Tableau 1** : Quelques types d'enregistrements stockés dans le DNS. On voit que des informations supplémentaires concernant chaque hôte peuvent être ajoutées comme MX et HINFO.*

L'espace des noms du DNS est structuré en arborescence. Pour une meilleure utilisation, des sous arbres peuvent être délégués à d'autres serveurs. Deux arbres logiques différents sont utilisés. Le premier fait correspondre les noms d'hôte tels que x.info.fundp.ac.be à des adresses telles que 138.48.32.y. Le second arbre correspond à la requête inverse et contient des enregistrements de type pointeur. Il n'y a pas forcément de relation entre ces deux arbres.

***Service** : DNS*
Corruption du DNS

Cette séparation peut poser certains problèmes. Un pirate qui contrôle une portion inverse de l'arbre de correspondance peut décider d'y insérer de fausses informations et donc le faire mentir. Le pirate va alors faire une tentative de Rlogin sur une machine qui, faisant confiance à l'enregistrement corrompu contenu dans l'arbre, acceptera la connexion.

La plupart des nouveaux systèmes se protègent contre cette attaque en effectuant une vérification croisée : après avoir retrouvé le nom supposé de l'hôte via DNS, ils utilisent l'arbre inverse pour retrouver l'ensemble d'adresses qui correspond. Si l'adresse utilisée pour la connexion ne se trouve pas dans la liste, la demande de connexion est rejetée.

Service : DNS
Variantes de la précédante

Il existe des variantes de l'attaque précédante qui peuvent être à l'origine de dégâts beaucoup plus importants. Dans une de ces variantes, le pirate contamine la mémoire cache du serveur des réponses DNS de la cible avant de lancer sa demande de connexion. Quand la cible fait la vérification croisée (expliquée plus haut), celle-ci réussit et la tentative de connexion du pirate également.

Service : DNS
Refus de service

Une autre variante consiste à inonder le serveur DNS de la cible avec de fausses réponses, le rendant incohérent.

Bien que les toutes dernières versions des software DNS soient immunisées contre ces types d'attaques, il serait cependant imprudent de croire qu'il n'existe plus de trous dans la sécurité.

Dans le cas générale, une authentification basée sur les adresses est préférable à une authentification basée sur les noms.

Service : DNS
Points vulnérables

Un intrus qui interférerait avec le déroulement propre d'un DNS pourrait lancer de nombreuses attaques comme, par exemple, l'attaque de refus de service ou la récolte de mots de passe. Il existe de nombreux autres points vulnérables.

Service : DNS
SEQUENCE NUMBER Attack

Dans certaines implémentations de résolveur DNS, il est possible de lancer une attaque SEQUENCE NUMBER contre un hôte particulier. Quand l'utilisateur cible tente de se connecter à une machine distante, un pirate peut générer une réponse du serveur de domaine correspondant à la requête de la cible. Pour mener une telle attaque il faut

cependant connaître à la fois le port UDP utilisé par le résolveur d'adresses du client et le numéro de séquence DNS utilisé pour la requête. Ce dernier est parfois assez facile à obtenir car certains résolveurs commence toujours leurs numéros de séquence à 0.

Service : DNS
Attaques combinées

Une attaque combinée sur le DNS et sur les mécanismes de routage peut être catastrophique. L'intrus peut virtuellement intercepter toutes les requêtes demandant de traduire un nom en adresse et remplacer l'adresse correcte par l'adresse d'une machine corrompue ; ceci peut permettre à l'intrus d'espionner tout le trafic et d'établir, s'il le désire, une liste assez complète de mots de passe.

Service : DNS
Corruption de la machine

C'est pour cette raison que les serveurs de domaines sont des cibles de choix. Un pirate suffisamment déterminé pourrait trouver utile de prendre le contrôle d'un serveur par d'autres moyens : en corrompant, par exemple, la machine sur laquelle se trouve ce serveur ou en interférant avec les liens physiques mêmes qui relie celle-ci à Internet.

Il n'existe pas de défense contre la première, ce qui suggère qu'un serveur de domaine ne devrait tourner que sur des machines extrêmement sûres. Quant au deuxième, on peut s'en protéger en employant des techniques d'authentification sur les réponses du serveur de domaine.

Service : DNS
Espionnage

Le DNS, même s'il fonctionne correctement, peut être utilisé à des fins d'espionnage. Le mode de fonctionnement normal du DNS est d'envoyer des requêtes spécifiques et de recevoir des réponses spécifiques. Cependant, il existe une requête de transfert de zone qui permet le chargement d'une section entière de la base de données ; en appliquant cela récursivement, un plan complet de l'espace des noms peut être obtenu. Cette base de données représente un risque potentiel au niveau sécurité ; si, par exemple, l'intrus sait qu'une certaine catégorie d'hôtes ou de systèmes d'exploitation est

particulièrement vulnérable, cette base de données peut alors être consultée pour trouver de telles cibles (Cfr. SINFO dans tableau 1). Les autres utilisations possibles de cette base de donnée incluent l'espionnage : le nombre et le type de machines dans une organisation particulière peut également permettre de jauger l'importance de celle-ci et les différentes ressources affectées à un projet particulier.

Heureusement, le DNS possède un code d'erreur pour *refused* ; c'est un interdit administratif contre de tels transferts de zone. Ce code devrait être employé pour les requêtes de transfert de zone (AXFR) provenant d'hôtes non répertoriés comme étant des serveurs secondaires légitimes. Malheureusement, il n'existe pas de mécanisme d'authentification disponible pour la requête de transfert de zone ; le mieux que l'on puisse faire étant l'authentification basée sur l'adresse.

Service : DNS
Trailing level

Il persiste une caractéristique disponible de beaucoup d'implémentations DNS qui peut se révéler dangereuse : elle autorise les utilisateurs à omettre les niveaux de trace (trailing levels) si le nom désiré et le nom de l'utilisateur ont des composants communs. Supposons, par exemple, que quelqu'un sur foo.info.fundp.ac.be tente de se connecter à une destination bar.com. Avant d'essayer bar.com (qui est correcte), le DNS essaiera d'abord bar.com.fundp.ac.be, bar.com.ac.be et bar.com.be. Cela comporte un risque. En effet, si quelqu'un crée un domaine com.be, il peut intercepter le trafic qui ne lui est pas destiné.

Les problèmes de l'authentification mis à part, le DNS est problématique pour d'autres raisons. Le DNS contient, en effet, un grand nombre d'informations à propos du site : noms et adresses des machines, système d'exploitation, structure de l'organisation, etc.

Protéger ces informations contre les petits curieux est une tâche ardue. Restreindre les transferts de zone à des serveurs secondaires est un bon début mais, cependant, un pirate intelligent peut chercher l'espace d'adresses exhaustif du réseau via les requêtes inverses du DNS, ce qui lui procure ainsi une liste des noms d'hôtes. A partir de là, il pourra étendre ses recherches et trouver d'autres informations utiles.

2.2.7 Le serveur d'authentification.

Comme alternative à l'authentification basée sur l'adresse, certaines implémentations utilisent le serveur d'authentification. Un serveur qui désire connaître l'identité d'un de ses clients peut contacter le serveur d'authentification de celui-ci et demander des informations sur un utilisateur employant une connexion donnée. Par définition cette méthode est plus sûre que la simple authentification basée sur

l'adresse car, ici, on utilise une seconde connexion TCP qui n'est pas sous le contrôle du pirate. Cette technique permet de mettre en échec les attaques SEQUENCE NUMBER et SOURCE-ROUTING. Elle conserve cependant certains risques.

Service d'authentification
Considérations générales

Tout d'abord, il est évident que tous les hôtes ne sont pas compétents pour exécuter des serveurs d'authentification. Si l'hôte client n'est pas sûr, connaître l'utilisateur n'a pas d'importance puisque la réponse n'est pas digne de confiance.

Service d'authentification
Attaque RIP

Ensuite, le message d'authentification lui-même peut être soumis à une attaque RIP. Si RIP a été utilisé pour altérer l'idée que se fait la cible de la façon d'atteindre un certain hôte, la requête d'authentification reposera sur les mêmes données de routage corrompues.

Service d'authentification
Variante de SEQUENCE NUMBER Attack

Finalement, si l'hôte cible est déconnecté, une variante de l'attaque SEQUENCE NUMBER peut être utilisée ; après que le serveur ait envoyé une requête d'ouverture de connexion TCP au serveur d'authentification présumé, le pirate peut compléter le séquence d'ouverture et envoyer une fausse réponse.

Service d'authentification
Refus de service

Un risque moins évident est qu'un faux serveur d'authentification peut toujours répondre « no ». Ceci constitue donc une attaque de refus de service.

Un serveur qui emploierait l'idée que se fait un autre hôte de l'utilisateur considéré utilisera un moyen de validation plus sûr[21]. TCP en lui-même est inadéquat.

2.2.8 Mécanismes permettant la recherche d'informations sur les utilisateurs d'un réseau.

2.2.8.1 Introduction.

Deux protocoles standards, *finger*[22] et *whois*[23] sont couramment utilisés pour rechercher des informations sur des individus. De ces deux protocoles, c'est l'emploi de *finger* qui est le plus dangereux.

2.2.8.2 Le service *finger*.

De nombreux systèmes implémentent un service *finger*. Ce service peut être utilisé pour collecter des informations sur des utilisateurs individuels ou sur ceux qui sont connectés au système actuellement. Le serveur affiche des informations intéressantes à propos des utilisateurs :

- leur nom complet;
- leur adresse électronique;
- leur numéro de téléphone;
- leur numéro de bureau;
- quand ce compte a été utilisé pour la dernière fois (les comptes peu ou pas utilisés possèdent généralement des mots de passe facile à deviner);
- d'où l'utilisateur s'est connecté pour la dernière fois (ce qui permet de localiser sa cible avec plus de précision pour une attaque indirecte);

> *Service : Finger*
> *Deviner les mots de passe*
>
> ***Malheureusement, de telles données apportent de la farine au moulin du cracker de mots de passe. Si un administrateur de réseau décide de fournir cette possibilité aux utilisateurs de son réseau, il doit le faire en ayant conscience qu'il répand ainsi des informations utiles.***
>
> ***La donnée la plus fréquemment utilisée est la correspondance qui est faite entre le nom entier de l'utilisateur et une adresse électronique. C'est pour cette raison que de nombreux sites sont réticents à supprimer finger.***

Un compromis raisonnable consiste à installer son propre démon *finger* qui contient une base de donnée allégée.

[21] Cfr. Algorithme de Needham-Schroeder.
[22] Harrenstein 1977.
[23] Harrenstein et White 1982.

Par exemple :

```
% finger sda@info.fundp.ac.be
[info.fundp.ac.be]

If you want to send mail to someone
at computer science institut,
address your mail using the
following format :
firstname.lastname@info.fundp.ac.be
```

Le protocole *whois*.

Le protocole *whois* est beaucoup plus bénin car il ne fournit que des informations de contact.

La portée des serveurs *whois* standards (à travers tout l'Internet) se trouvant à NIC.DDN.MIL et RS.INTERNIC.NET est limitée. Quelques organisations possèdent leur propre *whois*, mais cela n'est pas très courant.

Le courrier électronique.

Introduction.

Le courrier électronique est probablement le service qui apporte le plus de valeur à Internet. Cependant, il est assez vulnérable aux mauvaises utilisations. Quand il est implémenté normalement, le serveur de courrier ne fournit pas de mécanismes d'authentification. Ceci laisse la porte ouverte aux faux messages[1].

Le protocole POP (Post Office Protocole).

Le protocole POP permet à un utilisateur distant de consulter son courrier à partir d'un serveur central. L'authentification se fait à l'aide d'une seule commande qui contient à la fois le nom d'utilisateur et le mot de passe.

Protocole : POP
Mots de passes

Cependant, combiner les deux dans une seule commande réclame l'utilisation de mots de passe conventionnels. De tels mots de passe deviennent de moins en moins populaires ; ils sont trop vulnérables aux pirates qui enregistrent ce qui se passe sur les lignes de communication (wire-tappers) et aux dysfonctionnements intentionnels ou non, etc.

Comme alternative, de nombreux sites ont adoptés des mots de passe utilisables une seule fois (*one time password*). Dans le cas des mots de passe utilisables une seule fois, l'hôte et une machine valable pour l'utilisateur partagent une clé cryptographique. L'hôte doit lancer un challenge au hasard ; les deux côtés cryptent ce numéro, et

[1] *Voir RFC 822 pour l'utilisation d'un entête de ligne crypté. Voir RFC 1040 pour une discussion de nouveaux standards de cryptage pour le courrier électronique.*

l'utilisateur le retransmet vers l'hôte. Le challenge est aléatoire, la réponse est donc propre à chaque session, ce qui rend inactive l'écoute indiscrète (*eavedropping*). L'utilisateur ne connaissant pas la clé (elle est dissimulée dans la machine de façon à ne pas pouvoir être retrouvée), le mot de passe ne peut être transmis à une autre personne sans que celle-ci ne soit privée de la possibilité de se loger.

Dans POP version 3, on a divisé le nom d'utilisateur et le mot de passe en deux commandes. Cependant, on définit également un mécanisme optionnel de pré-authentification des connexions utilisant les mécanismes de Berkeley.

PCMAIL.

Le protocole PCMAIL utilise des mécanismes d'authentification similaires à ceux utilisés dans POP version 2.

Protocole : PCMAIL
Mot de passe

L'emploi de celui-ci est cependant plus dangereux car il supporte une commande de changements de mot de passe. Cette requête exige qu'à la fois l'ancien et le nouveau mot de passe soient envoyés non cryptés.

Les Firewalls

Introduction.

Comme vous avez pu le constater tout au long des chapitres précédents, Internet, comme toute autre société, est harcelé de petits plaisantins qui pratiquent l'équivalent électronique du *tag*, de la destruction de plis postaux ou encore de l'embouteillage volontaire des voies publiques. Cependant, certaines personnes utilisent Internet à des fin de travail, d'autres ont des données sensibles ou prioritaires à protéger. Le but d'un *Firewall* est de maintenir les petits malins en dehors du réseau où l'on désire travailler. Un firewall est un dispositif qui possède les caractéristiques suivantes : **tout** trafic de l'intérieur vers l'extérieur, et inversement, doit passer à travers le *Firewall, s*eul le trafic **autorisé** d'après la politique de sécurité est autorisé à passer.

La plupart des organisations et des centres de données traditionnels possèdent une **politique de sécurité** et des pratiques qui doivent être respectées. Dans le cas où la politique de l'entreprise dicte la façon dont les informations doivent être protégées, le *firewall* va jouer un rôle important dans la mesure où il va être le lieu de concentration de la politique collective en matière de sécurité.

Un *firewall* d'une organisation peut également servir, de par sa localisation, d'ambassadeur pour Internet. Beaucoup d'entreprises utilisent leurs *Firewall*s pour y

placer des informations concernant leurs produits et services, des fichiers à télécharger, la solutions à certains *bugs*, etc.

Politique de sécurité et *firewall*

Introduction et exemples.

Avant de se mettre à déployer un mécanisme de sécurité, il convient de se poser quelques questions d'ordre général pour définir une politique de sécurité:

Quelles sont les ressources que l'on doit protéger ?

Doit-on sécuriser des machines ou seulement des fichiers ou des données sensibles stockées sur celles-ci ? Doit-on protéger certaines machines ou toutes les machines ? Quelles valeurs ont les ressources que l'on désire protéger ? Quels sont les services que l'on va conserver (TFTP, FTP, SMTP,etc.) ?

Incidents[2]	**Nombre**
guest logins	296
rlogins	62
Ftp password	27
whois	10
SNMP	9
X11	8
TFTP	5
ARP	4
Nombre de sites sources	95

Ce tableau présente un exemple de liste d'incidents suspects et leur fréquence

Contre qui/quoi doit-on se protéger ?

Quel est le profil de l'agresseur ? Est-il seul ou fait-il partie d'un groupe ? Est-ce un *teenager* ou un professionnel ? Tout cela est bien entendu lié à la valeur des informations (mentionnées plus haut).

Nombre d'attaques	Source (extension DNS)
200	edu
38	com
6	gov
3	mil

Tableau présentant un exemple de sources DNS possibles pour des attaques.

[2] Tous les chiffres cités dans ce chapitre sont issus de « Firewalls and Internet Security » (Cheswick and Bellovin). Tous ces chiffres sont issus des *logs* qu'il on recueillis sur diverses périodes. Les paramètres exacts de leur implémentation et des périodes d'observation ne sont pas indiqués car ils n'apporteraient rien à la compréhension du problème.

Nombre d'attaques	Source (extension DNS)
19	ca
10	de
7	uk
5	au
3	nl
3	il

Tableau présentant un exemple de sources géographiques possibles pour des attaques.

Quel prix est-on prêt à payer pour la sécurité ?

Quelle est la valeur des informations détenues ? Quelles seraient les conséquences de leur altération, de leur destruction de leur vol ?

Le prix à payer pour se doter d'un *firewall* peut aller de 0$ (TIS ToolKit sur une machine existante) à plus de 50 000 $ (matériel + coût d'implémentation par des consultants (CISCO)).

Sécuriser un réseau c'est en fait sécuriser les informations sensibles véhiculées ou stockées par celui-ci.

Une fois que des réponses à toutes ces questions ont été trouvées, il convient alors de rédiger sa propre **politique de sécurité**.

Hypothèse de travail.

Dans ce travail, le concept de politique de sécurité est abordé dans le cadre de l'emploi d'un ***firewall*** pour se protéger de l'extérieur. Comme nous le verrons plus tard, le système de protection développé ici peut être répertorié dans les moyens de défense du type *firewall*.

Définition.

Une **politique de sécurité** est un ensemble de décisions qui, ensembles, déterminent la position d'une organisation vis à vis des problèmes de sécurité.

Plus précisément, une politique de sécurité établit **les limites du comportement acceptable** et **les réponses qui doivent être données aux violations**. Dans ce travail, nous n'aborderons la sécurité que sous son aspect de connexion avec le monde extérieur (Internet)[3].

La politique de sécurité est un concept qui varie d'une organisation à l'autre. Un département d'une université a des besoins différents par rapport à une entreprise commerciale ou à un site militaire. Dans le cas des facultés, par exemple, aucune

[3] L'aspect sécurité interne a déjà fait l'objet d'un précédent mémoire : « Dévellopement d'un outil de surveillance et de détection d'attaques sur réseaux Ethernet. » par Borhème Regaya.

mesure concrète n'existe en matière de sécurité en ce qui concerne la préservation contre les attaques externes au réseau.

Dans ce mémoire, nous nous intéresserons essentiellement aux limites de ce qui est acceptable du point de vue du comportement. C'est essentiel lors de la mise en œuvre d'un *firewall*. Les réponses qui seront apportées aux violations, elles, se feront au coup par coup et sous la responsabilité du responsable du réseau.

La **première étape** en matière de politique de sécurité consiste, bien évidemment, à faire le choix de se connecter ou non au réseau extérieur. En fait, toute politique de sécurité est une conséquence de la résolution par le responsable de la sécurité du dilemme suivant :

ne pas se connecter par peur des risques

↕

se connecter pour profiter des avantages du réseau

Toute connexion au réseau Internet doit donc se faire en pesant les avantages et les inconvénients, les risques et les bénéfices qu'une telle connexion pourrait apporter.

La **deuxième étape** se présente également sous forme d'un choix : faut-il décider de bloquer tout le trafic et ne laisser passer que ce qui est explicitement autorisé, ou bien, au contraire, vaut-il mieux tout laisser passer et bloquer seulement ce qui est explicitement interdit.

Dans notre cas, nous[4] avons décider de tout laisser passer et de ne bloquer que ce qui est explicitement interdit. Dans l'ouvrage de Cheswick et Bellovin[5], ceux-ci ont opté pour l'autre solution. Celle-ci est évidemment plus sûre. N'oublions pas qu'il ne s'agit ici que d'un début, que d'un test et que nous sommes dans une université et pas à l'armée.

Remarque : Une chose à ne jamais perdre de vue : tout programme est *buggé*, tout programme de sécurité possèdent donc des *bugs* de sécurité et donc aucun système de protection n'est jamais sûr à 100%.

Définitions

Définition générale

On définit un *Firewall* comme une liste de composants placés entre deux réseaux et qui respectent tous certaines propriétés :

- Tout trafic de l'intérieur vers l'extérieur, et inversement, doit passer à travers le *Firewall*.
- Seul le trafic autorisé d'après la politique de sécurité est autorisé à passer.
- Le *Firewall*, lui-même, doit être impénétrable.

[4] En accord avec le responsable du réseau des Faculté, Bruno Delcourt.

[5] W.R. Cheswick and S. Bellovin « Firewalls and Internet Security. Repelling the wily Hacker. »

Un Firewall classique est généralement sûr car il n'est pas accessible par rlogin, etc. Un Firewall ne doit pas être considéré comme un hôte à part entière ; son emploi devrait être exclusivement réservé à la sécurité.

Une caractéristique importante d'un firewall est qu'il fournit un point de passage unique où la sécurité et l'**audit** peuvent être concentrés.

3.3.2 Définition plus précise.

Un Firewall est constitué de différents composants. Ces différents composants sont repris sur la figure ci-dessous :

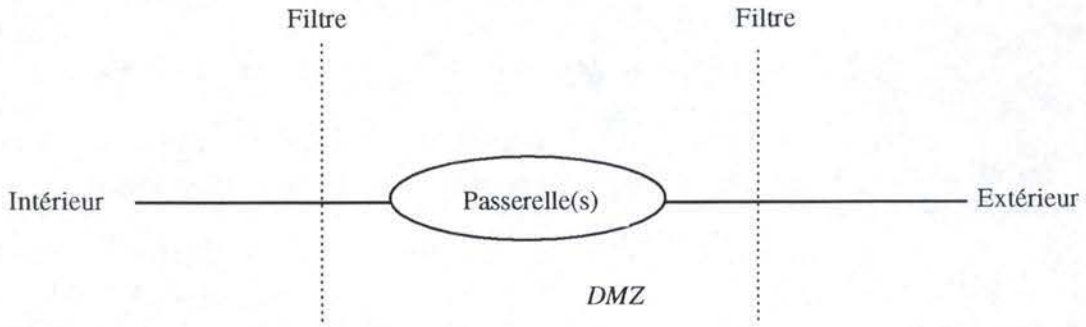


Schéma abstrait d'un Firewall typique.

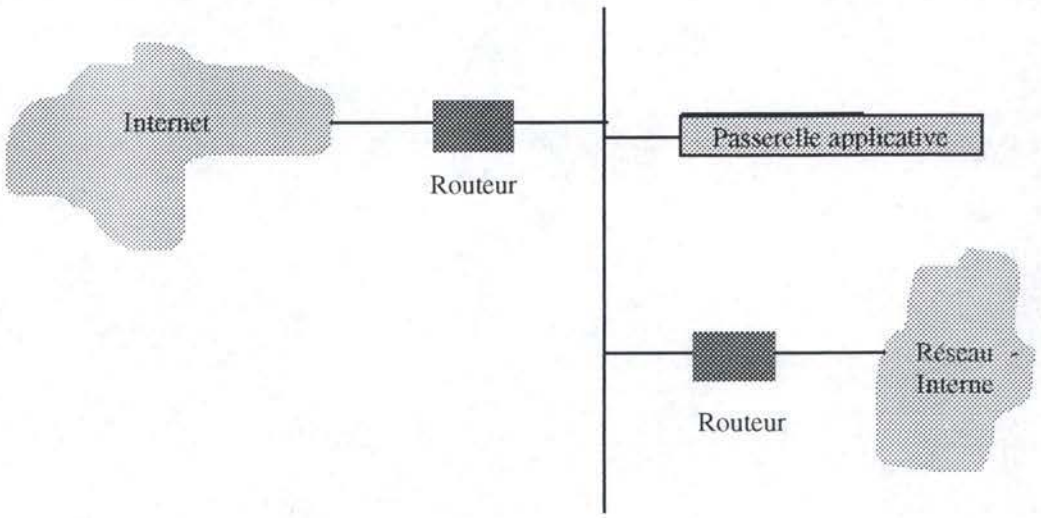


Schéma typique d'un firewall

Les filtres (aussi appelés écrans(*screens*)) bloquent les transmissions de certaines classes de trafic. Ces filtres sont généralement des routeurs. Une passerelle est une machine ou un ensemble de machines qui fournissent un relais qui permet de compenser les effets des filtres. La partie du réseau protégée par une passerelle est

habituellement appelé zone démilitarisée (*DMZ : DeMilitarized Zone*). Les passerelles de ce type sont parfois assistées par une passerelle interne. Le plus gros du trafic entre extérieur et intérieur se déroulant entre ces deux passerelles.

Les filtres, ou les passerelles peuvent être omis ; la présence ou l'absence de passerelle ou de filtre dépend de la configuration de firewall adoptée. Cette configuration peut varier d'un firewall à un autre.

En général, le filtre extérieur est utilisé pour protéger la passerelle des attaques externes. Le filtre intérieur, lui, permet de protéger l'intérieur contre une passerelle qui serait passée aux mains de l'ennemi. Une passerelle exposée au feu des attaques (c-à-d sans filtre externe protecteur) est aussi appelée HOTE BASTION.

On peut classifier les firewalls en trois catégories :

1° - les firewalls fonctionnant par filtrage de paquets (*packet filtering*).

2° - les firewalls basés sur des passerelles de niveau circuit (*circuit gateways*).

3° - les firewalls basés sur des passerelles de niveau application (*application-level gateways*).

Dans la pratique, ces trois types sont généralement mélangés.

3.3.3 Les coûts.

Un firewall n'est pas gratuit, les différents coûts engendrés sont :

- le coût du hardware et coûts de maintenace.
- le(s) coût(s) du(des) softwares et leurs maintenances.
- le coût des updates.
- le coût des formation du personnel.
- les coûts administratifs.
- le coût et les inconvénients engendrés par la coupure d'accès à certains services.

Ces différents coûts sont à opposés à ceux qui résulteraient de la mise hors service de machine suite à une attaque, ou à la perte d'informations sensibles (espionnage).

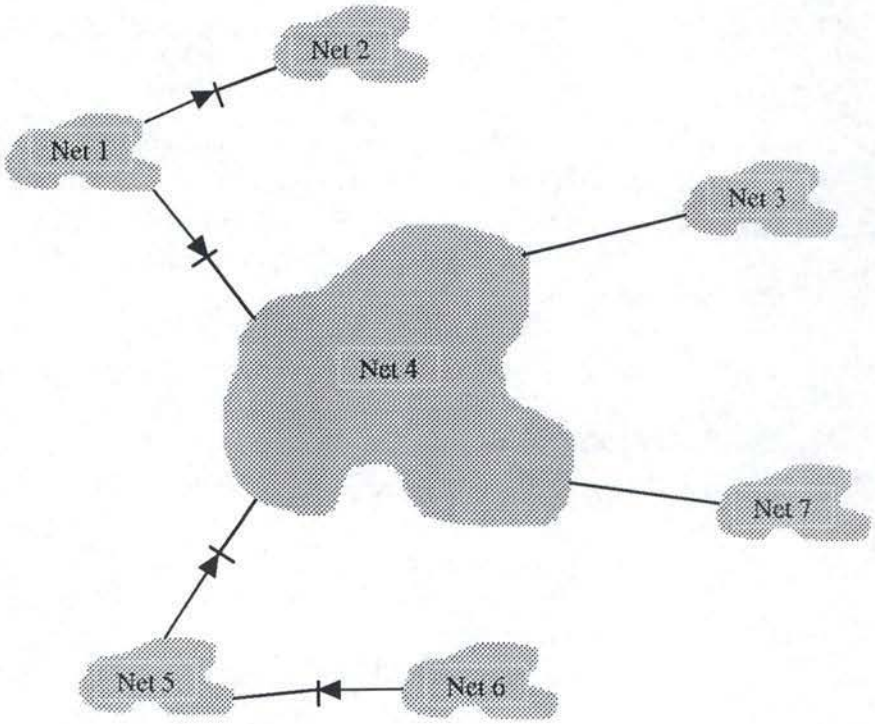
Les coûts résultant de l'emploi d 'un firewall dépendent fortement du type de site à défendre. Tout comme pour la politique de sécurité, on peut facilement admettre qu'une université n'a pas les mêmes objectifs en matière de protection qu'une organisation commerciale. Cette même organisation commerciale n'ayant pas les mêmes besoins qu'une organisation militaire. Une université considère généralement que l'accès libre à Internet contribue fortement à faire de celle-ci une communauté ouverte sur l'extérieur[35].

[35] Rappelons que 90% des hackers proviennent des milieux universitaires, c.-à-d. de ces milieux ouvert sur le monde.

Comment positionner un *Firewall*.

Traditionnellement, les *firewall*s sont positionnés entre le monde extérieur et l'organisation à protéger. Cependant, une grande organisation peut avoir besoin d'employer des *firewall*s internes pour isoler des sous-domaines de sécurité. Un **domaine de sécurité** est un ensemble de machines sous un contrôle administratif commun avec une **politique de sécurité commune** et ayant un même niveau de sécurité.

Considérons le schéma ci-dessous. Les différents domaines de sécurité sont représentés en grisés.



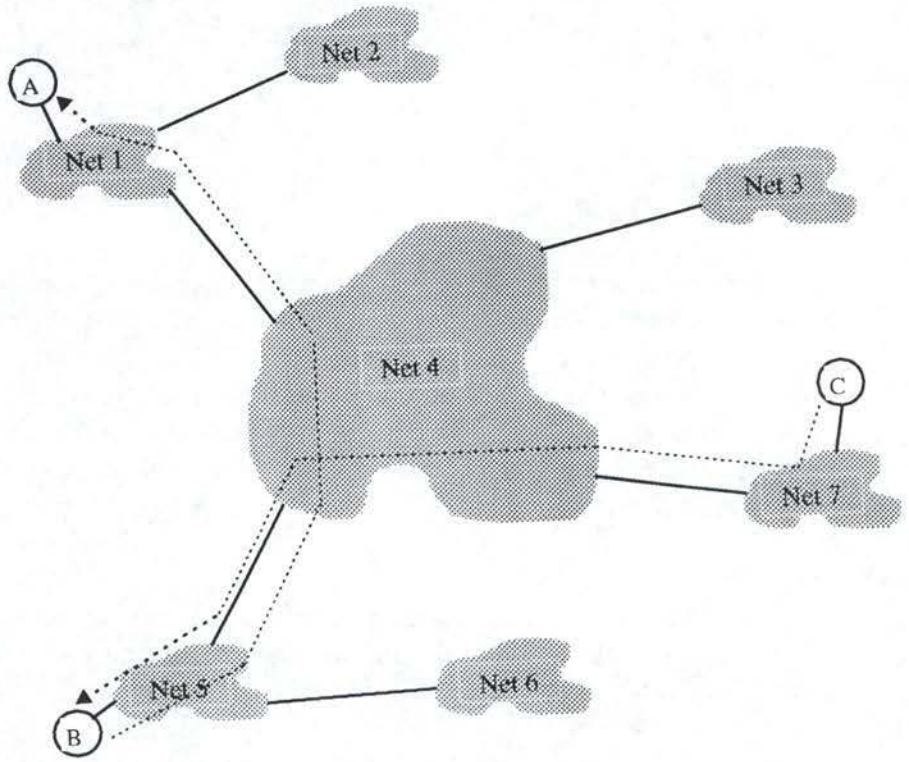
Les *firewall*s, réprésentés par des diodes (--|>--) devraient être placés aux frontières entre les différents domaines de sécurité. La flèche de la diode pointe vers les (potentiellement) méchants. Dans ce cas, on voit que le réseau Net 1 ne fait pas confiance (*trust*) à aucun autre réseau, même pas Net 2, alors que celui-ci est manifestement un réseau interne (lien exclusif avec Net 1).

Il existe de nombreuses raisons d'ériger des *firewall*s internes. Dans la plupart des grandes compagnies, les employés n'ont pas (et ne devraient pas avoir) accès toutes les informations du système. Certaines des informations disponibles ne sont, en effet, utiles qu'au développeur et au personnel de support technique et non à l'utilisateur de base.

Même les membres du personnel autorisé devraient passer par l'intermédiaire d'une passerelle de sécurité lorsqu'ils traversent un *firewall* ; sinon, si leurs machines, qui se situent en dehors du *firewall* (du côté interne) sont corrompues, l'équipement

particulièrement sensible qui réside à l'intérieur du *firewall* pourrait être la prochaine cible. Un *firewall* doit contrôler l'accès et la confiance à accorder à un autre hôte de manière prédictible.

La confiance transitive (*transitive trust*) est une configuration que l'on rencontre également sur les réseaux (voir figure ci-dessous). Supposons que la machine A, en plein accord avec la politique de sécurité locale, décide d'étendre sa confiance à la machine B. De même, la machine B décide d'étendre sa confiance à la machine C, en concordance avec sa politique de sécurité. Il en résulte que dorénavant la machine A fait confiance à la machine C, qu'elle le veuille ou non, qu'elle le sache ou non. Un *firewall* permet de se protéger contre ce genre de danger.



Possibilités et limites des *Firewalls*.

Les firewalls :

- **permettent** de concentrer tout ce qui concerne la politique de sécurité en un seul point.
- **permettent** de gérer les droits d'accès.
- **Sont l'endroit idéal** pour l'implémentation d'un système de monitoring et d'audit (c'est essentiellement de cette qualité dont nous allons essayer de tirer parti dans ce mémoire).
- **Permettent** de générer des alarmes.

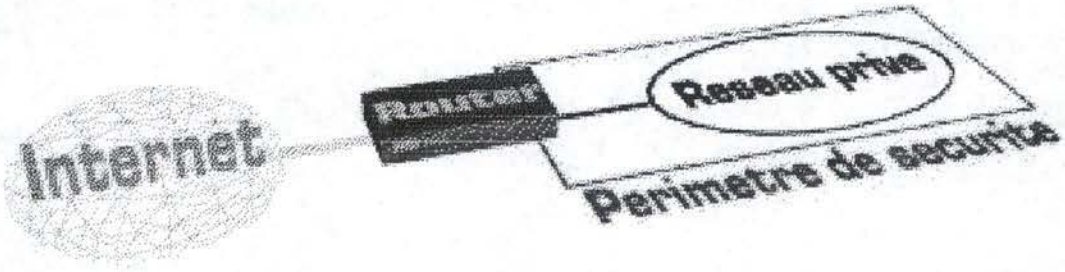
Cependant, les firewalls :

- **ne protègent pas** contre les attaques qui ne les traversent pas
- **ne protègent pas** contre les intrusions ou les attaques requérant un accès physique
- **sont inefficaces** pour lutter contre les virus et autre cheval de Troie.

Les trois catégories de *firewall*s.

Les *firewall*s basés sur la technique de filtrage des paquets (*packet filtering firewalls*).

Introduction.



Le filtrage de paquets est une solution bon marché et assurant un bon niveau de sécurité. Le faible coût est dû à l'existence des capacités de filtrage existant dans le software des routeurs eux-mêmes. C'est cette solution qui se rapproche le plus du cas traité dans ce mémoire. Les quelques points qui suivent retracent, à titre indicatif, les différents aspects du filtrage. Bien que le mode filtrage utilisé ci-dessous soit basé sur la politique du « Tout bloquer, Laisser passer ce qui est autorisé », les problèmes rencontrés ainsi que les solutions apportées n'en restent pas moins valides dans notre cas.

Les filtres de paquets fonctionnent en laissant tomber des paquets en se basant sur leur adresse et leur numéro de port source et destinataire. Le filtrage peut se faire à l'entrée, à la sortie ou des deux côtés à la fois.

En général, le contexte n'est pas considéré, c'est à dire que la décision est prise en fonction du contexte du paquet **courant.** Notons, ici, qu'avec ASAX, nous allons pouvoir faire intervenir le contexte : en effet, chaque paquet a une histoire et celle-ci va jouer un rôle dans la prise de décision.

C'est à l'administrateur que revient la tâche d'établir la liste des machines et des services autorisés et non autorisés. Il est facile de permettre ou d'interdire l'accès à ou depuis un hôte ou un réseau grâce au filtrage. On peut, par exemple autoriser les accès IP entre un machine A et une machine B et interdire tous les accès vers B en provenance de A.

La plupart des politiques de sécurité requièrent un contrôle un peu plus fort que cela : il faut définir l'accès à des services spécifiques pour des hôtes qui, par défaut,

seront considérés comme *untrusted*. Par exemple, on voudrait permettre à n'importe quel hôte de se connecter à la machine A, mais on désire seulement recevoir ou envoyer du courrier. Les autres services étant non autorisés. La technique de filtrage des paquets permet un contrôle de relativement bonne qualité. Cependant, cette technique est dangereuse et propice aux erreurs. Pour bien maîtriser cette technique, il faut une connaissance approfondie des protocoles UDP et TCP.

3.4.1.2 Configuration d'un filtre.

Pour bien configurer un *firewall*, il est conseillé de suivre les différentes étapes suivantes (cette procédure est valable quelque soit le mode de filtrage choisi):

1° Etablir une politique de sécurité globale c.-à-d. choisir entre tout bloquer et laisser passer ce qui est explicitement autorisé ou tout laisser passer et bloquer ce qui est explicitement interdit.

2° Etablir une politique de sécurité ponctuelle c.-à-d. choisir ce que l'on veut laisser passer et ce que l'on veut bloquer.

3° Spécifier formellement les paquets permis ou interdits en termes d'expressions logiques sur les différents champs des paquets.

4° Réécrire ces expression de manière à pouvoir les implémenter. Dans ce mémoire il va s'agir de programmer les règles correspondantes en langage RUSSEL.

3.4.1.3 Exemples utiles.

Supposons que notre politique de sécurité consiste à autoriser les mails vers l'intérieur (SMTP, port 25), mais seulement vers notre passerelle. D'autre part, les mails en provenance du site HACKSITE doivent être bloqués en raison de leur penchant à envoyer des fichiers de plusieurs mégabytes de données en une seule fois. Un filtre qui implémenter ce genre de règle pourrait ressembler à ceci :

Action	Notre hôte	n° de port	Leur hôte	n° de port	Commentaires
bloquer	*	*	HACKSITE	*	ne par faire confiance à HACKSITE
permettre	notre passerelle	25	*	*	connexion à notre port SMTP

Les règles sont appliquées du sommet vers le bas. Les paquets non explicitement autorisés sont rejetés. Ceci revient à dire que tout ensemble de règle est implicitement suivi par :

bloquer	*	*	*	*	bloquer tout par défaut

Cette dernière règle est la matérialisation de la politique de sécurité globale adoptée ici, « bloquer tout ce qui n'est pas explicitement autorisé ».

Remarquons que cet ensemble de règle ne nous protège en rien contre une attaque lancée à partir du port 25 sur la machine source.

Un meilleur ensemble de règles consiste à permettre les appels **sortant** vers un port 25. Nous voulons en fait permettre à nos hôtes de faire des appels vers les ports 25 d'autres hôtes extérieurs, c.-à-d. que nous voulons permettre les mails vers l'extérieur. Il est possible de faire la distinction entre les appels entrant et les appels sortant en considérant le champ CODE des paquets TCP[29]. En effet, avec TCP, même si les données ne vont que dans une seule direction lors d'une connexion, des paquets de contrôle et d'accusé de réception (ACK) voyagent dans l'autre sens[30]. Les paquets ayant leur bit ACK positionné sont des paquets faisant partie d'une communication en cours. Les paquets n'ayant pas ce bit positionné font partie de paquets de demande d'ouverture de connexion et ne doivent être acceptés que s'il sont en provenance d'un des hôtes internes. L'idée est qu'un étranger puisse continuer une conversation mais ne puisse pas en initialiser une[31].

Action	Notre hôte	n° de port	Leur hôte	n° de port	CODE	Commentaires
bloquer	{nos hôtes}	*	*	25	-	nos paquets vers leur port SMTP
permettre	*	25	*	*	ACK	leurs réponses

3.4.1.4 Le traitement de la fragmentation des paquets IP.

L'existence de la fragmentation des paquets IP rend difficile le filtrage des paquets. Exception faite pour le premier paquet, les autres paquets ne contiendront pas de numéros de port. Cependant, le premier fragment contient un numéro de port et sera donc traité normalement. S'il est rejeté, le paquet sera incomplet et donc finalement écarté. En ce qui concerne la fuite d'informations de l'intérieur vers l'extérieur, rien n'empêche deux personnes de prendre des conventions quant aux numéros de ports employés[32].

3.4.1.5 Les problèmes importants à résoudre.

Dans les paragraphes qui vont suivre, nous allons aborder les problèmes que l'on peut rencontrer avec FTP, X11, DNS, Telnet, Finger, Whois. Ces problèmes et leurs solutions sont intéressants à analyser dans la mesure où ils vont nous permettre d'éviter des pièges lors de l'implémentation avec ASAX.

3.4.1.5.1 Le filtrage de sessions FTP.

Comme nous l'avons vu, les fichiers transférés via FTP utilisent une connexion secondaire. Si le canal de contrôle à un serveur sur LEURHOTE utilise la connexion <notrehôte, notreport, leurhôte, 21>, le transfert de fichier se déroulera sur <notrehôte, notreport, leurhôte, 20>. De plus, le serveur doit initialiser l'appel de

[29] Les différentes valeurs possibles pour ce champ sont ACK, SYN, RST, PSH, URG, FIN.

[30] Par exemple, un paquet contenant une requête d'ouverture de connexion n'a pas son bit ACK positionné, les autres paquets ont ce bit positionné.

[31] On doit faire l'hypothèse raisonnable suivante : un paquet faisant partie d'une conversation en cours (ACK positionné) sera rejeté s'il n'a pas fait d'abord l'objet d'un paquet de demande d'ouverture de connexion.

[32] Avec screend (de DEC), tous les paquets appartenant à la même séquence de paquet subissent le même sort : soit écartés, soit acceptés. Il existe d'autres software de filtrage de paquets : AMU (de Texas), Karlbridge (pour PC),etc.

transfert de fichier. Nous avons donc le même problème que précédemment, mais sans avoir la possibilité de filtrer en se basant sur la direction de l'appel.

On peut tenter de résoudre le problème en utilisant la gamme de nos numéros de port pour prendre une décision[33]. La plupart des serveurs, et donc la plupart des cibles d'attaques, résident sur ports de numéros faibles ; la plupart des appels vers l'extérieur ont tendance à utiliser des numéros de ports élevés (>1023).

Action	Notre hôte	n° de port	Leur hôte	n° de port	CODE	Commentaires
permettre	{nos hôtes}	*	*	*	-	nos appels sortant
permettre	*	*	*	*	ACK	réponses à nos appels
permettre	*	*	*	>1023	-	trafic vers des non-serveurs

Les paquets peuvent passer si :

- ils proviennent d'une machine interne.
- ce sont des paquets de réponse à une communication initialisée sur une de nos machines.
- ils sont destinés à des ports de numéros >1023 sur nos machines.

Cependant, tous les appels aux serveurs internes ne sont pas coupés car tous les serveurs ne sont pas localisés sur des numéros de port faibles (par exemple, X11 voir plus loin). Ce problème peut se résoudre par l'emploi de la commande PASV vers le serveur, lui commandant une ouverture passive de connexion et permettant ainsi un appel sortant à travers le *firewall* pour le canal de données. En temps normal, le client choisit un numéro de port au hasard et l'envoi au serveur par l'intermédiaire de la commande PORT; le serveur ouvre alors une connexion vers ce port. Avec PASV, le client émet une commande PASV; le serveur choisit alors un numéro de port au hasard et demande au client d'y initialiser la connexion.

3.4.1.5.2 Le filtrage de sessions X Window.

Le problème de X11 est en partie semblable à celui de FTP : son utilisation requiert l'emploi d'un appel entrant. En fait, le terminal de l'utilisateur (écran, clavier, souris) est le serveur; les applications X11 se connectent à celui-ci via TCP. Si les applications sont exécutées sur un hôte extérieur, la connexion au serveur implique un appel venant de l'extérieur, ce qui est bloqué par l'ensemble de règles prescrit ci-avant. C'est un véritable problème car c'est une catégorie d'applications qui devrait être typiquement permise[34].

Des connexions X11 non autorisées représentent une réelle menace car des intrus peuvent lire des données à partir de l'écran de l'utilisateur, écouter les frappes clavier et même, dans certaines circonstances, générer des entrées clavier *buggées*.

[33]Certains sites préfèrent construire eux-mêmes leur version de FTP en relation avec une SOCKS library. SOCKS est un système proxy générique qui peut être compilé sur la partie cliente d'un protocole pour que ce protocole puisse travailler à travers un firewall.

[34]Un hôte interne désirant bénéficier de facilités externes.

Ce danger peut être géré par une bonne coopération entre utilisateurs et par une bonne configuration des serveur X11. Il vaut en effet mieux être trop prudent que pas assez et donc prendre la décision de bloquer tous les appels entrant vers des ports de numéro 6000 à 6100[35].

Cependant, si on prend la décision de bloquer tout le trafic vers ces ports, plutôt que juste les appels entrant, on prend le risque de déranger des programmes clients qui s'exécutent également dans cette gamme.

3.4.1.5.3 La maîtrise du DNS.

Le problème de DNS réside dans la sensibilité même des informations qu'il contient. Certaines organisations désirent maintenir cachés vis-à-vis de l'extérieur les noms contenus dans le DNS. Tout dépend de la politique de sécurité adoptée en la matière.

L'approche présentée ci-dessous permet de cacher les noms d'hôtes du réseau Internet vis-à-vis d'Internet. Le succès de cette approche réside dans le fait que le client DNS d'une machine n'a pas à converser avec un serveur DNS sur cette même machine. Autrement dit, le fait qu'il existe un serveur DNS sur une machine n'est pas en contradiction avec le fait de rediriger l'activité des clients DNS de cette machine vers le serveur DNS d'une autre machine (c'est même souvent avantageux).

Tout d'abord, il faut installer un serveur DNS sur un hôte bastion avec lequel le monde extérieur peut converser. Il faut installer ce serveur de telle façon qu'il clame avoir autorité comme serveur DNS pour le domaine à protéger. En fait, tout ce que ce serveur doit savoir, ce sont les informations que le réseau interne est d'accord de communiquer à l'extérieur : les noms et adresses des passerelles, les enregistrements MX, etc. Appelons ce serveur le serveur public.

Ensuite, il faut installer un serveur DNS sur une machine interne. Ce serveur clame également avoir autorité comme serveur DNS pour le domaine interne; cependant, contrairement au serveur public, celui-ci dit la vérité. Ce serveur est le serveur de nom interne normal dans lequel on place toutes les informations contenues habituellement dans un serveur DNS. Il faut également programmer ce serveur pour qu'il redirige les requêtes qu'il ne peut résoudre vers le serveur public[36].

Finalement, il faut programmer les clients DNS[37], y compris ceux sur le machine contenant le serveur public, pour qu'il utilisent le serveur interne.

Un client interne émettant une requête concernant un hôte interne s'adresse au serveur interne pour obtenir une réponse; un client interne émettant une requête concernant un hôte externe s'adresse au serveur interne qui, à son tour, demande au serveur externe qui se renseigne sur Internet et renvoie la réponse au serveur interne. Un client sur le serveur externe fonctionne de la même façon. Par opposition, un client

[35]S'il y a plus de 100 serveurs X11 en service sur une machine, il faut élargir la gamme des ports à protéger.
[36]Utiliser « forwarders » dans /ect/named.boot sur Unix.
[37]Fichier /etc/resolv.conf sur Unix.).

externe émettant une requête concernant un hôte interne recevra une réponse « restreinte » du serveur public.

Cette approche suppose l'existence d'un *firewall*, basé sur le filtrage des paquets, positionné entre ces deux serveurs et leur permettant d'entretenir des conversations du type DNS entre eux. Ce filtre permet également de restreindre les échanges DNS parmi les autres hôtes.

Une autre chose qu'il peut être utile d'employer dans cette approche, ce sont les champs PTR dans les domaines IN-ADDR.ARPA. Ceci permet à un hôte externe d'obtenir le message « unknown.domaine.interne » plutôt qu'un message d'erreur. Ceci satisfait certains sites FTP anonyme qui nécessitent absolument la connaissance du nom de la machine à laquelle ils sont en train de parler. Cependant cela peut échouer dans les sites où une vérification DNS croisée est d'application entre les noms et les adresses d'hôtes et inversement.

Cependant, il faut noter que cette approche ne permet pas d'éviter la fuite des noms d'hôte par l'intermédiaire des entête de E-mail ou par des articles publiés par des membres du personnel du réseau à protéger.

3.4.1.5.4 Le filtrage de sessions Telnet.

Dans le cas de Telnet, la simple configuration d'un routeur pour permettre des connexions sortantes suffit. Tout comme pour FTP, on établit une liste de règles de filtrage au niveau du port Telnet (port 23).

3.4.1.5.5 Le filtrage de Finger et Whois.

La solution est de permettre des connexions au port Finger (port 79) seulement à partir de machines de confiance. Les requêtes Finger devant respecter le formalisme suivant : finger user@host.domain@*firewall*

Cette approche ne fonctionnera qu'avec la version standard de Finger sur Unix. Certains serveurs Finger ne permettent pas de requête de la forme user@host@host.

De nombreux sites bloquent les requêtes Finger entrantes pour un tas de raisons: pour les bugs que présentait Finger dans le passé (rendus célèbres par l'Internet Worm), pour les risques que représente la publication de ces informations sensibles, etc.

3.4.1.6 Les protocoles sans adresse fixe.

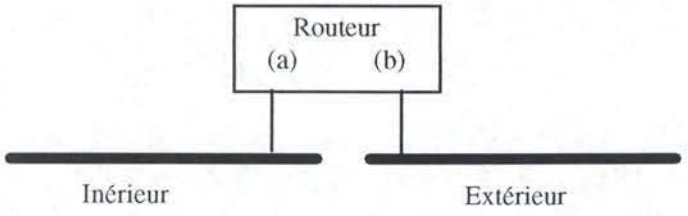
Certains services sont problématiques à traiter au moyen du filtrage des paquets car ils n'utilisent **pas toujours** de numéro de port fixe. Certains n'utilisent d'ailleurs **jamais** de numéro de port fixe.

Deux exemples de ce genre de protocoles sont TCPmux et Portmapper[38]. Dans les deux cas, les programmes clients contactent un programme de *mapping* plûtot que l'application. Portmapper exécute également des requêtes d'enregistrement en provenance d'applications, les informant de leurs numéros de port courant. TCPmux, d'autre part, appelle l'application directement, en lui passant la connexion ouverte.

La solution, avec TCPmux, peut consister à contrôler l'accès au port TCPmux. Cela peut aller de la libre circulation au blocage complet. Avec Portmapper, tous les services ont leur propre port. La seule solution consiste donc à bloquer l'accès à tous les ports possibles utilisés par les serveurs de type RPC, ce qui n'est pas facile à faire.

3.4.1.7 Positionnement du filtre.

Le filtrage de paquets peut s'effectuer en plusieurs endroits. La figue ci-dessous montre un routeur typique. Les paquets peuvent être examinés au point (a),au point (b), ou aux deux points à la fois. De plus, les filtres peuvent être appliqués aux paquets entrant, sortant, ou aux deux. Tous les routeurs ne permettent pas toutes ces variantes; le style de filtre utilisé en sera affecté.



Un routeur firewall typique.

Filtrer les paquets à la sortie du routeur peut améliorer l'efficacité car la recherche de la règle de filtrage à appliquer et son application peuvent être combinés avec le parcours de la table de routage. Cependant, certaines informations ont étés écartées, comme le câble physique par lequel le paquet est arrivé ;ce qui peut être important dans la détection des attaques fonctionnant sur base d' address-spoofing. Filtrer à l'entrée du routeur peut le protéger lui-même d'éventuelles attaques.

Strictement parlant, il existe un troisième choix en ce qui concerne le positionnement du filtre : ne pas se soucier de l'interface. Le filtre *screend* se comporte de cette façon. Toutes les décisions sont prises sur base uniquement des adresses contenues dans les paquets; c'est cette solution que nous allons envisager dans ce travail. Remarquons que cette technique ne nous permet pas de nous protéger contre l'attaque address-spoofing. Dans le cadre de ce mémoire, nous ne sommes pas menacer par ce type d'attaque. En effet, Belnet nous protège pour ce qui est de l'extérieur.

Il faut prendre garde également à ce que le *firewall* applique les règles du filtre dans l'ordre où elles ont été écrites. L'ordre d'exécution des règles est en effet un facteur déterminant pour la qualité de détection d'un *firewall* (voir exemple ci-avant).

[38]utilisé par SunOs pour RPC.

3.4.1.8 UDP et le filtrage de paquets.

Filtrer des circuits TCP est difficile. Filtrer des paquets UDP propre à une fonctionnalité n'est rien moins qu'impossible. La raison réside dans la différence essentielle existant entre TCP et UDP : TCP est un protocole de circuit virtuel (l'établissement de ce circuit se faisant dans un certain contexte); UDP est un protocole de datagramme où chaque message est indépendant. Comme nous l'avons, filtrer TCP requiert l'utilisation du champ CODE (et plus particulièrement du bit ACK) pour distinguer les appels entrant et les réponses à des appels sortant. UDP ne possède pas de tel indicateur; on est donc obliger de se fier au port source qui peut faire l'objet de falsification.

Un exemple simple illustre le problème : supposons qu'un hôte interne désire émettre une requête vers le serveur echo UDP d'une machine externe. Le premier paquet devrait contenir

<localhost, localport, remotehost , 7>

où localport est dans la gamme de numéro de port élevés. La réponse serait

<remotehost, 7, localhost, localport>

et le routeur ne saurait pas si localport était une destination sûre. Un paquet entrant

<remotehost, 7, localhost, 2049>

est probablement une tentaivie de subversion du serveur NFS interne. Pire encore, les services de type RPC utilisent des numéros de port dynamiques, parfois dans la gamme des numéro de port élevés.

Une prise de position conservatrice consiste à bannir tout appel sortant employant le protocole UDP. Ce n'est pas que les requêtes soient dangereuses, c'est plutôt que l'on ne peut pas se fier aux réponses. La seule exception que l'on peut faire concerne les protocoles basés sur une relation un à un. Un bon exemple est NTP (*Network Time Protocol*). Le déroulement normal d'une opération s'effectue en effet sur les ports numéro 123 (source et destination). On peut donc admettre les réponses sans danger puisque celles-ci s'effectuent à un numéro de port fixe[39].

3.4.1.9 Avantages et inconvénients des firewalls basés sur le filtrage des paquets.

En plus des avantages et des inconvénients propres à tous les firewalls (cités plus haut), les filtres de paquets ont les particularités suivantes :

Avantages :

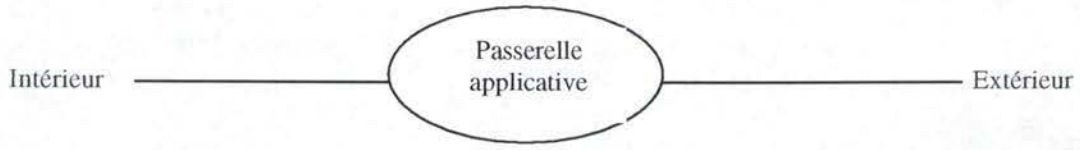
- Coûts faibles.
- Transparents pour les applications et pour les utilisateurs.
- Mise en place rapide.

[39]Remarquons que lorsqu'un machine *reboot* elle ne demande jamais son temps à l'extérieur (via NTP). Cet exemple n'a donc pas de raison d'être dans des conversations avec un hôte externe.

Inconvénients :

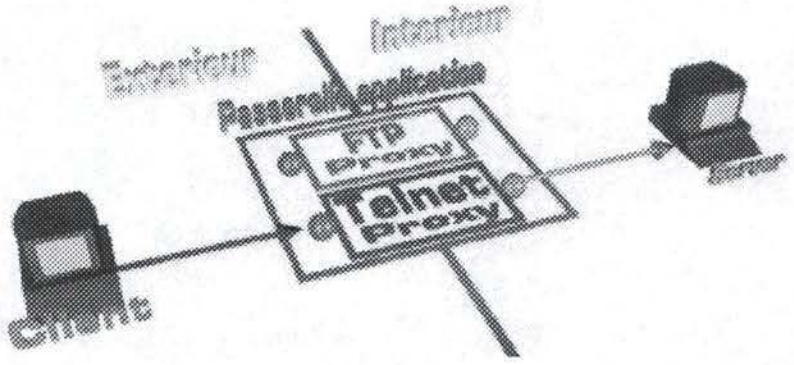
- Complexité d'élaboration des règles de filtrage.
- Plus le nombre de règles augmente, plus la performance du filtre diminue.
- Impossibilité de prendre le contexte en considération.

3.4.2 Les *firewall*s passerelles de niveau application (*Application-Level Gateways*).



Comme nous venons de le voir, le service de routage du trafic peut être implémenté à un niveau IP via l'utilisation de routeur filtrant le trafic en se basant sur un ensemble de règles. Il peut également être implémenté à un niveau application via l'utilisation des services et des passerelles proxy.

En fait, il faut choisir entre exposer une machine sur le réseau extérieur et y exécuter des services proxy pour Telnet, FTP, news, etc, ou installer un routeur filtrant les paquets et permettant la communication avec une ou plusieurs machines du réseau interne. Ces deux approches ont des avantages et des inconvénients : avec les machines proxy, on a l'assurance d'un **meilleur niveau d'audit** et donc un niveau de sécurité potentiellement plus élevé mais pour des **coûts de configuration plus importants** et une diminution dans le niveau de service qui peut être fourni[40].



Un serveur proxy (aussi appelé passerelle applicative ou forwarder) est une application qui régule le trafic entre un réseau à protéger et Internet. Les proxies sont souvent utilisés à la place des contrôleurs de trafic basés sur un routeur pour empêcher le trafic de passer directement entre deux réseaux. De nombreux proxies contiennent des extra *loggin* ou des supports supplémentaires d'authentification. Les proxies doivent **comprendre** le protocole d'application utilisé et peuvent également implémenter la sécurité spécifique à ce protocole (par exemple, un proxy FTP devrait être configurable pour permettre les paquets FTP entrant et bloquer les FTP sortant).

[40]En effet, un proxy différent doit être développé pour chaque nouveau service que l'on désire implémenter.

Les serveurs proxy sont spécifiques à un application. Si l'on désire supporter un nouveau protocole via un proxy, celui-ci doit être spécialement développé.

3.4.3 Les *firewall*s passerelles de niveau circuit (*circuit gateways*).

Une passerelle au niveau circuit est un simple relais de connexions TCP sans manipulation ou filtrage de paquets. Elle relaie la connexion à travers le *firewall* sans faire d'examens supplémentaires, de filtrage ou de gestion de protocole. Elle agit comme un fil, copiant telles quelles les données de l'extérieur vers l'intérieur (ou inversement). Cependant, les connexions semblent être originaires du *firewall*, ce qui a pour effet de cacher complètement les informations concernant le réseau protégé.

Ce genre de passerelles est souvent utilisée pour les connexions sortantes lorsque l'administrateur a confiance en ses utilisateurs.

Le principal avantage est qu'un Bastion Host peut être configuré comme une passerelle hybride supportant des services proxy pour les connexions en entrées, et des fonctions du niveau circuit pour les connexions en sorties. Cela rend l'utilisation du *firewall* plus facile d'emploi pour les utilisateurs qui désirent un accès direct à Internet, tout en permettant une protection efficace contre les attaques de l'extérieur.

Notons enfin que c'est l'endroit idéal pour le déploiement d'un Network Adress Transaltor comme une protection très efficace contre le spoofing, les adresses internes étant inconnues de l'extérieur.

4 Asax

4.1 Introduction

Après avoir survolé de façon plus ou moins approfondie ce qui peut être fait en matière d'attaque sur les réseaux et après avoir vu les principes de fonctionnement des *firewalls*, ce chapitre va être consacré à l'aspect « **audit** » de la sécurité. L'étude des fichiers de *log* va être réalisée à l'aide d'un programme appelé ASAX. L'analyse portera sur des paquets IP captés sur le réseau des facultés selon une topologie particulière permettant les tests avant implémentation.

4.2 Présentation d'ASAX

Les chapitres précédents ont permis de mettre en évidence la vulnérabilité des systèmes informatiques actuels face aux différents types d'attaques réseaux. Il est impossible de prévoir toutes les attaques possibles sur un système. On ne peut donc pas être assuré d'un niveau de sécurité absolu (Niveau A du DOD[41] par exemple). A l'heure actuelle, la conception d'un système fiable contre toute attaque est difficile à réaliser. En effet, pour cela il faudrait donner une formalisation de tous les mécanismes d'accès permettant de prouver mathématiquement l'impossibilité d'une utilisation non autorisé ou accidentelle. Ce type de formalisation n'est pas encore

[41] Voir annexes.

possible à implémenter étant donné la complexité des systèmes d'exploitation et des protocoles de communication, etc.

Les systèmes munis d'un haut niveau de sécurité comportent des mécanismes de stockage d'informations échangées avec le réseau. Ce sont les fichiers de traces (aussi appelés *audit trail* ou *log*). Ces fichiers sont analysés et la détection des attaques est exécutée a posteriori. Cette analyse permet de déceler des scénarios d'utilisations malicieuses, des tentatives d'intrusions, des virus, etc. après leurs productions (ou même en cours de productions).

ASAX est né d'un projet entre Siemens Nixdorf (Rhisne) et l'Institut d'Informatique des facultés universitaires Notre Dame de la Paix. Le projet ASAX *(Advanced Securiry Audit-trail Analysis on unix)* a pour objectif d'analyser des fichiers de traces provenant de tous horizons, il décrit également un langage général pour de telles analyses. **Ce langage est basé sur des règles** et est appelé RUSSEL *(RUle-baSed Sequence Evaluation Language).*

Le logiciel ASAX **permet l'analyse de n'importe quel fichier séquentiel en un seul passage**. Il permet entre autre l'analyse des différents fichiers de traces générés par un OS. Il fournit à l'utilisateur un langage d'interrogation simple qui lui permet d'exprimer ses requêtes d'analyse sur le fichier concerné. Cependant le fichier à analyser doit être mis sous un format standard NADF (*Normalized Audit Data Format*), simple et reconnu par ASAX.

Une des caractéristiques d'ASAX est son architecture qui est construite sur un outil d'analyse universel permettant l'adaptation du format de n'importe quel type de fichiers de traces. Une autre caractéristiques est le langage RUSSEL utilisé pour exprimer des requêtes sur ces fichiers. Ce langage est basé sur des règles qui sont bien adaptées à l'analyse des fichiers séquentiels en un seul passage.

Les outils d'analyse existants ne permettent pas d'analyser les fichiers de traces générés par les différents mécanismes d'audit. Ces mécanismes produisent des fichiers dont le format de l'enregistrement est différent du format exigé par l'outil d'analyse. ASAX permet de trouver des solutions à ces problèmes.

Pour être un outil puissant d'analyse, ASAX devrait affronter plusieurs types de problèmes liés à :

- la disparité des scénarios de violation de la sécurité;
- la grande quantité de données qui se trouve dans les fichiers de trace;
- la réutilisation des fichiers de traces et des outils d'analyses;
- la convivialité de l'interface fournit au gestionnaire de sécurité[42].

Rappelons qu'il existe deux grand types de violations de sécurité : les pénétrations **internes** et les pénétrations **externes** au réseau. C'est à cette dernière catégorie que nous nous intéressons dans ce travail.

[42] Ce point ne fait pas partie du mémoire mais serait peut-être intéressant à développé.

Nous pouvons classifier les approches de l'analyse des fichiers de traces en deux groupes:

- les approches basées sur une statistiques d'une activité normale moyenne des utilisateurs, processus, ou de tout le système etc.
- les approches basées sur les connaissances des experts concernant les prédictions d'intrusion.

Le premier type d'approches est approprié à la détection des scénarios d'intrusion connus, alors que le deuxième type est plus approprié pour détecter des scénarios inconnus.

L'outil d'analyse des fichiers de traces devient parfois un système expert dont le rôle est non seulement de détecter les violations de sécurité, mais aussi d'améliorer sa propre connaissance en mettant à jour le profil de quelques utilisateurs, et en ajoutant les scénarios de quelques nouvelles attaques.

Un autre facteur important qui rend l'analyse encore plus difficile est relié à la taille des fichiers de traces. Si nous ne faisons pas plusieurs filtrages importants et des sélections adéquates, le système d'exploitation pourrait produire une très grande quantité de données d'audit qui rend l'analyse impossible. Le système d'exploitation permet une sélection à partir de la source mais pour éviter la production de fichiers de traces inutilisables, cette sélection doit être très adéquate. L'audit intelligent doit ainsi fournir une sélection adéquate et appropriée. Cette sélection peut être exécutée au niveau de la source, dans ce cas elle est appelée une **pré-sélection**, ou bien au niveau de l'analyse, on parlera alors d'une **post-sélection**. En pratique, une phase de filtrage de pré-traitement qui utilise des règles simples pourrait soulager l'outil d'analyse en écartant les grandes quantités de données non pertinentes.

Un autre problème concernant la quantité de données à analyser est relié au choix entre l'analyse *"on-line"* et l'analyse *"off-line"*. L'analyse est *"off-line"*, si elle est exécutée après la création des fichiers de traces, c'set à dire *a posteriori*. Cependant, et sous certaines conditions, l'évaluation peut également s'effectuer pendant la création de ces fichiers, et nous parlerons d'analyse *"on-line"*. Ces deux cas seront étudiés dans la troisième partie de ce mémoire.

4.3 Les objectifs d'ASAX.

Le but d'ASAX est de permettre une analyse universelle, efficace et puissante des fichiers de traces.

L'universalité se base sur son architecture. La réutilisation dans ASAX concerne le niveau physique. L'objectif principal étant de fournir un outil universel d'analyse des fichiers de traces qui tourne sur une grande gamme de systèmes d'exploitation. La raison de ce choix est double:

- permettre l'analyse de tous les fichiers de traces fournis et qui sont préalablement traduits en un format approprié (NADF);

- assurer le maximum d'efficacité qui reste l'objectif majeur de l'analyse des fichiers de traces.

En pratique, on définit un format de fichier d'audit normalisé (NADF pour *Normalized security Audit Data Format)* qui doit être assez flexible pour que tous les fichiers de traces existants puissent être traduits facilement. L'analyse est ainsi exécutée sur des fichiers de traces normalisés.

4.4 Les problèmes principaux

Le problème de la taille des fichiers de traces constitue une contrainte assez importante car la gestion de l'espace disque pourrait devenir critique pour l'officier de sécurité. Le temps CPU est aussi vulnérable à cette contrainte. Deux approches peuvent être retenues pour faire face à cet inconvénient:

1° La présélection;
2° La compression du donnée.

L'objectif étant donc de réduire la taille de ces fichiers de traces, cependant ceux-ci restent suffisamment importants: nous ne pouvons pas dire a priori quelles sont les informations qui vont être utilisées pour l'analyse. Ce problème peut être résolu selon que nous choisissons un nombre raisonnable de paquets à capter, ou que l'on opte pour une analyse *on-line* avec des options bien appropriées.

D'autres problèmes persistent. Ce sont par exemple:

- la variété de scénarios dans la sécurité;
- la réutilisabilité de l'outil de l'analyse;
- l'interface utilisateur.

4.4.1 La variété de scénarios dans la sécurité

Les scénarios dans la sécurité des systèmes informatiques prennent, comme nous l'avons vu, diverses formes. La classification des différentes formes de menaces a pour but de définir des catégories de scénarios détectables par l'évaluation des fichiers de traces. Néanmoins, dans chaque classe, beaucoup de scénarios d'attaques sont possibles, dont certains restent toujours imprévus. La difficulté provient donc du fait que l'on doit connaître a priori un scénario d'attaque et interroger le fichier de trace normalisé. Cela suppose une expertise non seulement des scénarios d'attaques mais aussi de la façon dont ils sont représentés dans le fichier.

4.4.2 La réutilisabilité, la généricité et l'universalité

La réutilisabilité d'un logiciel est une qualité très recommandée. Un logiciel peut être réutilisé dans différents environnements, matériels et logiciels, avec un effort d'adaptation minimal. Dans le cas d'outil d'analyse de fichiers de traces cette qualité est très importante vue la diversité des systèmes d'exploitation et de leurs versions multiples. Deux approches sont à envisager pour atteindre la réutilisabilité:

1. la conception d'un outil d'analyse paramétrisé et dans lequel les formats des différents fichiers de traces peuvent être instanciés. Il s'agit d'un outil "générique";

2 ou, un outil plus général permettant d'analyser tout type de fichiers de traces après conversion préalable en un format approprié à l'outil d'analyse: nous parlerons d'outil "universel".

4.5 Les qualités d'ASAX

ASAX est un outil de sécurité. Il permet d'analyser tout fichier séquentiel. Il a été conçu pour atteindre les objectifs suivants:

- analyser en un seul passage de longs fichiers séquentiels de sécurité des systèmes, mais aussi le traitement d'autres applications tels que l'analyse de flux de données via un réseau, etc.

- accepter toutes requêtes d'analyse possible y compris les requêtes qui portent sur la relation entre plusieurs enregistrements.

Les caractéristiques principales de l'évaluateur ASAX sont: **l'universalité,** la **puissance, l'efficacité** et la **portabilité.** Elles sont développées dans les sections suivantes.

4.5.1 L'universalité

Les systèmes d'exploitation munis d'un mécanisme d'audit génèrent des fichiers de traces de formats différents. L'objectif est de fournir un outil d'analyse universel qui pourrait tourner sur une large gamme de systèmes d'exploitation. Cette universalité est réalisée par le choix et la définition d'une structure de fichier souple et standardisée, appelé NADF.

L'analyse s'effectue uniquement sur des fichiers aux formats standardisées. Les fichiers de traces doivent donc être traduits en fichier de format NADF.

Vu la flexibilité du format NADF, la conversion de tout type de fichiers de traces en un tel format se ferait plus facilement avec la réalisation d'un adaptateur de format. Un des objectifs de ce mémoire est d'implémenter un adaptateur de format pour les fichiers de traces générés par le système.

En pratique, le format NADF est suffisamment flexible pour que tous les fichiers de traces existants puissent être traduits de façon simple. L'analyse de ces fichiers de traces est alors exécutée sur les fichiers normalisés. L'adaptateur de format devrait être fourni pour traduire chaque fichier de traces en un fichier de format NADF.

La faisabilité d'une telle approche a été démontrée par l'implémentation de formats adaptateurs pour les fichiers de traces générés par les systèmes d'exploitation BS2000 et SINIX de Siemens.

Le simple format des fichiers NADF garantit la portabilité au niveau physique. L'expérience avec BS2000 et SINIX montre que l'implémentation de tels adaptateurs de formats est assez simple.

4.5.2 La puissance: Le langage RUSSEL[43]

La puissance d'ASAX repose sur le langage de règles appelé RUSSEL *(RUle-baSed Sequential Evaluation Language)* qui permet à l'utilisateur d'exprimer ses requêtes d'analyse. Il s'inspire du principe des langages d'intelligence artificielle basé sur des règles avec chaînage avant.

RUSSEL est un langage de règles conçu spécifiquement pour l'analyse des fichiers de traces, et d'une façon plus générale, pour celle de n'importe quel fichiers séquentiel. Ce langage sera utilisé pour reconnaître des champs particuliers dans les fichiers séquentiels et ainsi déclencher des actions appropriées telles que l'activation d'une alarme, l'envoi d'un message etc. L'idée de RUSSEL est de prendre l'avantage de cette utilisation particulière et rendre le langage aussi efficace et facile à utiliser que possible.

RUSSEL peut être considéré comme un langage procédurale qui emploi des structures de contrôle pré-définies qui soient adaptés au raisonnement sur les séquences d'enregistrement. Ces structures de contrôle sont basés sur un mécanisme de déclenchement de règles qui peut être résumé comme suit:

1. le fichier de traces normalisé ou NADF est analysé enregistrement par enregistrement d'une façon séquentielle au moyen d'un ensemble de règles. A chaque instant, l'enregistrement courant est analysé et l'ensemble des règles est actif;

2. la règle active encapsule les connaissances essentielles concernant le passé de l'analyse. Cette connaissance est par la suite fournie à l'enregistrement courant sur lequel nous allons exécuter les règles. Le traitement génère à son tour de nouvelles règles qui vont être appliquées ultérieurement;

3. selon le point de vue du programmeur, une règle est considérée comme une procédure paramètrée classique mais qui peut entraîner des déclarations d'un type particulier d'actions: le déclenchement de règles;

4. pour déclencher une règle il faut fournir les paramètres réel pour la règle et spécifier le moment de déclenchement de cette règle: pour l'enregistrement courant, l'enregistrement suivant ou pour la fin de l'analyse;

[43] La syntaxe et la sémantique de langage RUSSEL se trouve en annexes.

5. le processus est initialisé par un ensemble de règles activé pour le premier enregistrement.

Pour programmer une requête dans le langage RUSSEL, il faut écrire un nombre de règles adéquates qui seront activer conformément au plan de contrôle général indiqué ci-dessus. Ces règles respectent une syntaxe particulière. Cette syntaxe et la sémantique du langage RUSSEL sont présentées en annexes.

Le fait que les fichiers de traces sont triés de façon chronologique joue un rôle essentiel dans l'approche qui stipule que l'analyse de ces fichiers consiste généralement en une recherche des sous-séquences chronologiques parmi tous les enregistrements d'événements.

Une déclaration de règles dans le langage RUSSEL consiste en un nom, un ensemble de paramètres formels, un ensemble de variables locales et une partie action. Une règle active est une instantiation d'une déclaration de règle avec des paramètres réels. Les règles actives sont exécutées une à la fois (à tout instant, il n'y a qu'une seule règle exécutée). Les structures de contrôle classique peuvent être utilisées pour écrire des actions conditionnelles, des actions répétitives, des actions séquentielles, etc. Les actions élémentaires comprennent des affectations, des déclenchements de règles et des routines d'appel externes. La dernière action permet au programmeur d'utiliser des procédures écrites dans un langage externe, comme le langage C, pour la lecture, l'écriture, l'accès aux structures de données complexes, etc.; les routines externes peuvent être appelées pour exécuter tous les caractéristiques qui ne concernent pas le traitement séquentiel du fichier de traces.

Les actions sont généralement appliquées sur quelques enregistrements spécifiques (Ex: ceux correspondant à des événements donnés, concernant un sujet donné, etc.).

Une règle est explicitement redéclenchée pour analyser l'enregistrement suivant, par défaut elle meurt après son déclenchement. L'objectif étant d'avoir un contrôle simple et directe sur la durée de vie des règles et pour éviter des mécanismes complexes qui tuent les règles dépassées.

La présentation du langage RUSSEL ci-dessus montre que ce langage est assez puissant pour supporter toutes les requêtes nécessaires qui concernent les fichiers séquentielles des traces. Cependant, le langage peut paraître difficile comme il utilise des styles différents de programmation. Le programmeur devrait donc raisonner en tenant compte de la structure de contrôle entraînée par le mécanisme du déclenchement des règles. Cette complexité semble être acceptable pour deux raisons:

- en comparaison avec d'autres langages moins efficace, RUSSEL ne semble pas être plus difficile dans son utilisation. Dans le langage RUSSEL, une approche pragmatique est considérée: l'efficacité des langages procéduraux (ce qui est reconnu) est bien exploitée. Cependant, les règles peuvent être écrites ou lues de façon déclarative. Nous pouvons ainsi dire que ce langage offre un bon compromis entre l'efficacité et l'aspect déclaratif de ses règles.

- l'idée principale du projet ASAX est de fournir un langage puissant, portable et implémentable de façon très efficace. Cependant, RUSSEL ne peut pas être considéré comme le langage final de l'utilisateur avec lequel les responsables de sécurité pourront construire leurs requêtes, mais seulement comme un langage intermédiaire.

4.5.3 L'efficacité: l'implémentation

L'efficacité d'ASAX se base sur son implémentation. Cette efficacité est un aspect critique de l'analyse des fichiers de traces à cause de la grande quantité de données à analyser. L'efficacité nécessaire est accomplie par les conceptions principales d'ASAX:

1 l'analyse du fichier de traces standardisé se fait en un seul passage séquentiel du fichier. A chaque instant, toute information sur le passé de l'analyse (c-à-d les enregistrements déjà analysés) est encapsulée dans l'ensemble des règles actives. Cette information est représentée par la valeur des paramètres effectifs de ces règles;

2 le traitement d'un enregistrement courant est précédé d'un pré-traitement (pour une raison d'optimisation du temps d'accès) permettant un accès direct à ses champs lors de l'évaluation de la condition de sélection. Après avoir évalué le *record* courant, un ensemble de règles actives est appliqué à l'enregistrement suivant;

3 évaluation efficace des conditions: les conditions qui figurent dans les règles sont évaluées de façon très efficace. L'évaluateur utilise une technique permettant de tronquer l'évaluation d'une condition dès que la valeur de vérité de celle-ci n'est pas vérifiée, sans devoir évaluer l'entièreté de la condition. Une telle efficacité d'évaluation des conditions est indispensable: cela correspond à une grande partie du temps CPU nécessaire à l'analyse.

Le premier principe est rencontré par le langage RUSSEL qui est spécialement conçu dans le but de permettre à toutes les requêtes d'être traitées en un seul passage. L'information nécessaire concernant le passé de l'enregistrement courant est encapsulée dans un ensemble de règles actives. En plus, comme RUSSEL supporte des appels de routine externe, les informations concernant le passé peuvent être également rangées dans les structures de donnée C et accédées via les routines externes.

4.5.4 La portabilité

L'évaluateur est implémenté dans un langage de haut niveau (le langage C), ce qui le rend facile à porter sur d'autres machines. En effet, la portabilité de cet analyseur est effective sur les systèmes d'exploitation Sinix, Unix et Dos.

Actuellement, ASAX tourne avec succès sur MX2, MX300, Sun Sparc (dans le cadre du mémoire, la portabilité d'ASAX sur les Sun s'est accomplie sans difficulté majeure) et aussi bien sur PC DOS.

4.5.5 Autres caractéristiques d'ASAX:

ASAX va fournir un adaptateur de format paramètrisé qui réalise la fonction de l'adaptateur du format, en commençant par une description déclarative du format du fichier de traces natif. Il est à noter que l'architecture ASAX ne fait pas de suppositions préalables sur le choix d'une analyse *"on-line" ou "off-line"*.

4.6 Le format NADF et les adaptateurs de format

ASAX ne reconnaît que les enregistrements sous le format NADF. Pour être donc capable d'analyser les fichiers d'**audit**, il faut d'abord passer par un traitement sur les données originales (qui pourrait être sous n'importe quel format). Pour cette traduction, on utilise un adaptateur de format *(format adaptator)*.

De plus, tous les types de données qui peuvent être trouvés dans le fichier transformé doivent être décrits dans un fichier de description des données *(data description file)*. En résumé, pour préparer le fichier de données qui doit être analysé, il faut fournir

1. un adaptateur de format,
2. un fichier de description des données.

4.6.1 Format NADF

Le format NADF (Normalized Security Audit Data Format) a été spécialement conçu pour les objectifs que recherchait ASAX. Dans ce format, chaque enregistrement consiste en la longueur réelle (sur 4 bytes) de l'enregistrement i.e. la longueur du contenu de l'enregistrement et les 4 bytes du champ longueur, le contenu de l'enregistrement lui-même.

Le contenu de l'enregistrement consiste en une séquence de champs. Un champ est représenté par :

- son identifiant (entier de 2 bytes)
- sa longueur (entier de 2 bytes)
- sa valeur (type fixé par l'identifiant)

Chaque champ a son nom lié à l'identifiant par une relation 1 - 1. La longueur du champ est en fait la longueur de sa valeur.

Real length of record		
id_1	$lgth_1$	$value_1$
...	...	...
id_i	$lgth_i$	$value_i$

...	...	...
id_n	$lgth_n$	$value_n$

Représentation d'un enregistrement au format NADF

La séquence de champs est triée selon le champ identifiant pour augmenter la performance du traitement de l'enregistrement au format NADF.

Le codage exact de ces champs n'est pas utile car il dépend de la machine sur laquelle l'analyse est exécutée. On peut seulement dire que id_i et $lgth_i$ sont entiers mais leurs représentations sont dépendantes de la machine. Dans RUSSEL, $value_i$ est traitée comme une chaîne de bytes.

4.6.2 Les adaptateurs de format

Les adaptateurs de format s'occupent de la traduction des enregistrements en entrée sous le format NADF.

il en existe deux formes

- *off-line* (sous la forme de programmes localisés hors du kernel ASAX)
- *on-line* .

Les premiers attendent en entrée l'ensemble des données à transformer et génèrent le fichier NADF correspondant. L'avantage de cette forme est de transformer le fichier d'entrée une et une seule fois et de garder le résultat de la transformation dans un fichier NADF qui peut être alors directement analysé.

Les trois routines I/0 quant à elles, supportent l'ouverture (routine *vopen),* la fermeture *(vclose)* du fichier à analyser et la lecture *(vread)* d'un enregistrement avec la transformation de cet enregistrement sous le format NADF incluse. L'avantage de ce type d'adaptateur est qu'il évite la génération de fichiers NADF. La transformation est ainsi intégrée à l'analyse, ce qui signifie qu'elle est réalisée progressivement, enregistrement par enregistrement, avec l'opération de lecture.

Une donnée audit est simplement transformée en ses identifiant, longueur et valeur.

Il faut savoir que la valeur de l'enregistrement au format NADF conserve la forme codée de l'enregistrement original (pour des raisons de sécurité, d'impossibilité de décodage, ...). La tâche de l'adaptateur de format ne se limite donc pas à une traduction pure et simple, celui-ci s'occupe également de conserver les règles de *mapping* entre la représentation interne et externe de la donnée audit.

4.7 Les fichiers de description de données

Un fichier de description de données *(data description file)* est un fichier qui contient la description de l'information qui peut être trouvée dans le fichier à analyser (sous sa forme NADF évidemment).

C'est un fichier texte séquentiel contenant un ensemble de descriptions de données précédé de quelques lignes d'en-tête.

Optionnelles, les six lignes d'en-tête fournissent de l'information générale. Une description de données représente la description d'un champ en 5 lignes

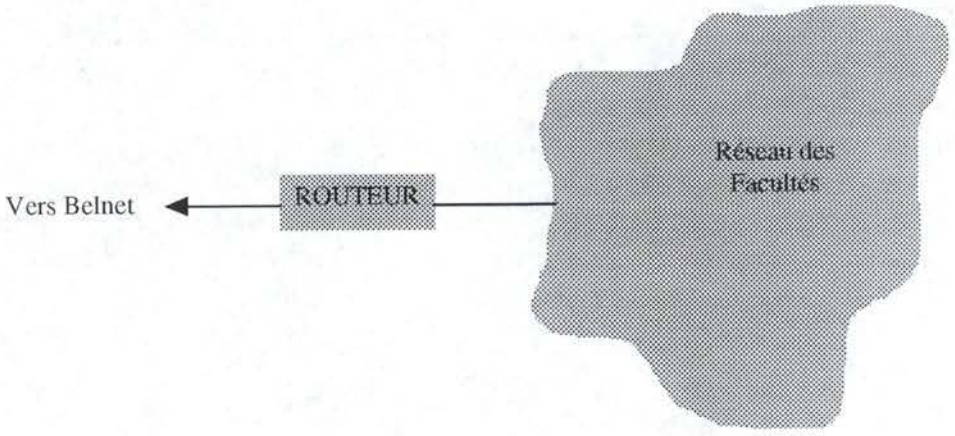
1.l'identifiant (qui sera l'identifiant sous le format NADF) ,
2.le type d'origine,
3.le type cible,
4.le nom (qui sera la référence à cette donnée dans un programme d'analyse),
5.la description sémantique (optionnelle).

Dans la troisième partie vous pourrez trouver le fichier de description de données ainsi que les adaptateurs de format off-line et on-line réalisés pour ce mémoire.

5 Présentation du réseau des Facultés.

Le réseau des Facultés comprend un peu plus de 1200 machines. Il s'agit d'un réseau de classe B dont les adresses correspondent à **138.48.x.y**. Celles-ci sont réparties sur 16 sous-réseaux, chacun d'eux correspondant plus ou moins à une faculté (**138.48.32.w**). Le réseau ne possède qu'un seul routeur. Ce routeur est connecté par une ligne à 256 Kbps à Internet par l'entremise de Belnet.

Schématiquement nous avons donc la situation suivante[44] :



[44] Tout cela n'est pas tout à fait exact : les sous-réseaux sont divisés en deux groupes, chaque groupe étant relié à un switch. Ce sont les deux switchs qui sont reliés au routeur.

L'idée principale de ce mémoire, est de pouvoir tester les possibilités d'ASAX en matière de protection des réseaux sur un cas réel. Ce cas réel nous est fournit par le réseau des facultés.

Il ne s'agit pas, à proprement parler, de bâtir un firewall qui filtrerait par exemple les paquets sur base de leurs adresses. Il s'agit plutôt d'utiliser les principes du firewall pour **détecter** les attaques, en être averti immédiatement et pouvoir ensuite prendre des mesures. Cela permettra ainsi de préserver le caractère ouvert que se doit de maintenir une université.

Pour cela, une machine, que nous appellerons le « sniffeur » va être placé à la frontière du réseau. Cette machine sera « à l'écoute » des paquets qui passent sur cette ligne unique que constitue la liaison avec Belnet. Idéalement, cette machine ne devrait pas posséder de connexion au réseau (autre que la possibilité de lire sur la connexion externe) afin d'éviter que cette machine soit elle-même l'objet d'une attaque qui la rendrait inutile.

C'est donc sur cette machine que va être placé ASAX afin de permettre l'audit de certains schémas d'attaques ou comportements qu'il est souhaitable de contrôler, ces schémas d'attaques et comportements étant programmé en RUSSEL[45].

6 Récapitulatif

Dans cette partie, nous avons donc vu qu'un réseau peut être attaqué de bien des manières. Nous avons également présenté deux solutions possibles à ces problèmes de sécurité : l'**audit** et les ***firewalls (filtres de paquets)***. Nous allons voir maintenant quelles sont les inconvénients de ces deux méthodes et comment une combinaison de ces deux méthodes pourrait nous mener à l'élaboration d'une solution plus performante.

6.1 Inconvénients de l'audit classique

Les réseaux sont généralement hétérogènes : les fichiers d'audit classiques peuvent être générés par de très divers systèmes d'exploitation au sein du même réseau.

> *Ici, on a l'avantage de travailler avec des datagrammes IP. Ces datagrammes font partie de la suite de protocole de télécommunication TCP/IP. Comme tout protocole, il impose des formats standards. Le format des paquets sera donc le même quelque soit le système d'exploitation à l'origine de l'expédition du paquet. Ensuite, ASAX étant programmé en C, il est implémentable relativement facilement sur la plupart des types de machines. La seule chose qui reste véritablement contraignante, c'est la réalisation de l'adaptateur de format. Nous verrons dans la troisième partie de ce mémoire que la programmation de celui-ci reste cependant relativement simple.*

[45] Les requêtes RUSSEL réalisées se trouvent dans la troisième partie de ce mémoire.

Baisse de performance des machines subissant l'audit : la machine sur laquelle est installé le programme de monitoring va subir une baisse de performance due à une certaine consommation de temps CPU.

Ici, seule une machine verra décroître ces performances : le sniffeur.

Les fichiers de *log* et les programmes d'audit doivent eux-mêmes être protégés : les programmes et fichiers sensibles peuvent, en effet, être la première cible du pirate qui commencera par désamorcer les processus d'audit avant de pratiquer une attaque ou qui effacera les fichiers de *log* juste avant de quitter.

Si, comme nous l'avons dit, le sniffeur est une machine qui ne fait que lire sur le câble sans véritablement faire partie du réseau, il n'y a aucun risque de le voir pénétré par un pirate[46].

Détection d'attaque différée : l'analyse se fait sur des fichiers de trace ; la détection d'une attaque ne peut donc se faire qu'une fois que celle-ci a eu lieu (et peut-être depuis longtemps) donc *a posteriori*.

Ici, avec l'emploi d'ASAX, nous avons l'énorme avantage de pouvoir déclencher un message d'avertissement dès qu'une attaque est détectée. Pour autant, bien sûr, qu'on utilise la version on-line *de l'adaptateur de format.*

6.2 Inconvénients des *firewalls*

Ne protègent pas contre les attaques qui ne les traversent pas : s'il existe d'autres accès au périmètre de sécurité que le *firewall*, celui-ci ne servira à rien.

Si, comme nous l'avons dit, il n'existe qu'une seule ligne permettant d'accéder au réseau des facultés à partir de l'extérieur. L'entièreté du trafic à destination ou en provenance d'Internet doit donc obligatoirement passer sur la ligne où se trouve le sniffeur.

Complexité d'élaboration et de gestion des règles de filtrage : rappelons que l'ordre dans lequel les règles sont écrites a une très grande importance.

Ici, il ne s'agit plus de programmer des règles de filtrage (bien que cela puisse se faire) mais bien des schémas d'attaques ou de comportements

[46] Pour autant que l'accès physique à cette machine soit sécurisé (mots de passe, etc).

suspects. Les requêtes permettant de formaliser les schémas de comportements ou d'attaques, sont rédigées en RUSSEL. RUSSEL est un langage qui se rapproche plus des langages de haut niveau que la programmation des listes d'accès d'un routeur.

Absence de contexte pour les paquets : le filtrage ne peut se faire que de manière non contextuelle, sur base des caractéristiques du paquet courant.

Ici, avec l'emploi d'ASAX, nous avons l'énorme avantage de pouvoir tenir compte du contexte. Nous allons en effet, à l'aide des requêtes RUSSEL pouvoir programmer des règles qui vont nous permettre de détecter un schéma d'attaque, c'est à dire tout une suite de paquets ayant des caractéristiques bien précises.

Inefficaces pour lutter contre les virus et autre cheval de Troie : ce genre de détection doit se faire à un autre niveau.

Ce n'est pas au niveau réseau qu'il faut essayer de détecter un virus, mais plutôt au niveau de l'application elle-même. Ce type de protection n'est pas du ressort de ce mémoire. Notons cependant qu'un mémoire précédant à permis de démontrer que l'on peut également utiliser ASAX pour la détection des virus.

De tout ce qui précède, on voit clairement que l'on devrait pouvoir réaliser une solution d'un très bon niveau de sécurité puisque la combinaison configuration-en-firewall/emploi-d'ASAX nous permet d'éliminer des défauts majeurs des outils de sécurité habituels.

Cette à cette tâche que va être consacrée la troisième et dernière partie de ce mémoire.

Troisième Partie : Réalisation Concrète.

1 Introduction

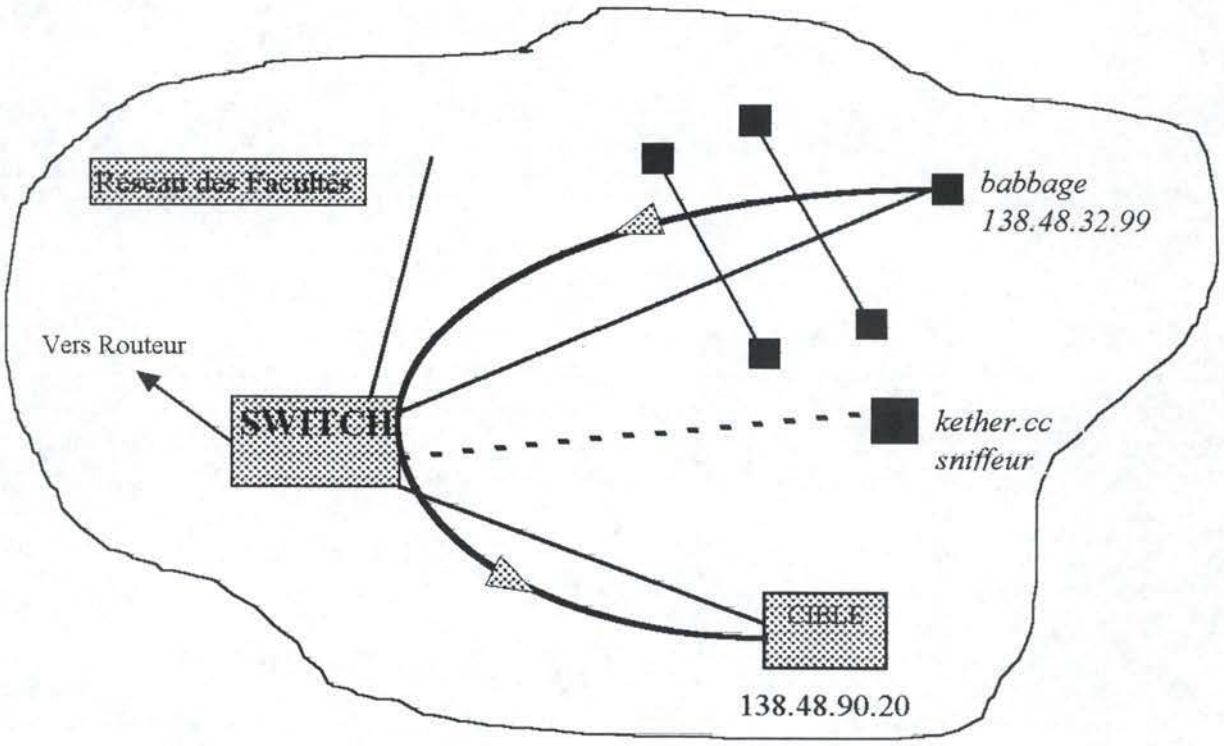
La première partie de ce mémoire était consacrée aux prérequis nécessaires à la compréhension des mécanismes d'attaques. La deuxième partie était consacrée, elle, aux éléments de sécurité importants à considérer dans le cadre de ce mémoire : les attaques, les firewalls, ASAX, la description du réseau des facultés et une justification de la configuration et des éléments employés pour réaliser l'outil de sécurité.

Cette partie, la troisième et dernière, va être consacrée à l'outil de sécurité lui-même : conception, implémentation, problèmes, tests, etc.

2 Configuration utilisée pour les tests

L'objectif premier de ce mémoire est donc la réalisation d'un outil de sécurité, utilisant ASAX pour la détection des schémas d'attaques. L'idée est d'implémenter cet outil au centre de calcul des Facultés afin de permettre la détection d'attaques et de comportements suspects dont on désire être averti. Il ne s'agit donc pas, dans ce mémoire, de concevoir un outil qui automatise la prise de mesures à l'encontre de ces attaques et de ces comportements. Il s'agit simplement de générer des messages d'avertissement à l'attention du responsable du réseau qui décidera alors des mesures à prendre.

Concrètement, avant d'implémenter réellement l'outil, il est nécessaire d'effectuer des tests afin de vérifier que les schémas d'attaques que l'on désire détecter sont bien ceux que l'on pense. Pour la réalisation des ces tests, la configuration est la suivante : pour simuler des paquets en provenance de l'extérieur, un nouveau sous-réseau a été créé. Sur ce réseau, on a placé une seule machine d'adresse *138.48.90.20*. C'est sur cette machine que vont être simulées les attaques et c'est sur la ligne qui mène à celle-ci que devrait être placé le *sniffeur* destiné à supporter les différentes évolutions de l'outil de sécurité. De cette manière, on va donc pouvoir simuler une attaque externe à partir de l'intérieur même de l'institut et, par exemple, à partir de *babbage*. Le *sniffeur*, comme *babbage* et la station *138.48.90.20* sont toutes des machines supportant le système d'exploitation du genre UNIX. Dans un premier temps c'est sur *babbage* que tournera ASAX , c'est donc la version d'ASAX adaptée à UNIX qui sera utilisée.



En fait, le sniffeur est connecté au switch qui relie ce nouveau sous réseau aux autres. Dans une configuration idéale, le sniffeur ne doit pouvoir avoir accès aux paquets qu'en lecture pour éviter d'être lui-même attaqué. Le sniffeur ne possède donc pas d'adresse IP puisque l'accès est en lecture seule[1].

3 Adaptateurs de format

3.1 Introduction

Comme nous l'avons vu, il est nécessaire de modifier le format des enregistrements pour pouvoir utiliser ASAX. Dans cette partie, nous allons voir quels sont les formats sources, quel est la forme du format NADF que l'on doit obtenir et les différents adaptateurs (*on-line*, *off-line*) conçus.

3.2 TCPdump

La lecture sur le câble se fait à l'aide TCPdump. TCPdump est un des meilleurs outils de monitoring disponibles pour système UNIX. Bien que son usage principal soit dédié à l'analyse de protocoles, il permet également l'enregistrement des paquets passant sur le câble. De la même manière, il permet de limiter le nombre de paquets enregistrés ; TCPdump possède un langage relativement riche permettant de spécifier le type de paquet qui doit être enregistré.

[1] C'est une hypothèse de travail ; en réalité, kether est une machine du réseau avec une adresse IP 138.48.4.35. Pour construire un système valable il faudrait cependant changer cela. N'oublions pas qu'il s'agit ici de l'ébauche d'un outil et pas d'un système complet et robuste.

Le produit brut généré par TCPdump ne peut être traité directement pour des raisons de performance pour la détection d'intrusion. Par construction, il n'y a pas de fichier de sortie en mode ASCII et, de plus, il peut y avoir de nombreuses conversations simultanées[2], entrelacées dans le fichier de sortie.

Dans ce mémoire, nous utilisons TCPdump pour lire les paquets passant sur le câble. TCPdump a été configuré de façon à ne pas prendre en considération les paquets de données d'une connexion TCP, seuls les paquets d'ouverture et de fermeture de connexion sont considérés ici.

En pratiquant de cette façon, on contribue déjà à la diminution du nombre de paquets destinés à être analysés par ASAX. Ci-dessous, un exemple de fichier de sortie pour une « conversation » telnet.

```
18:12:55.788512 138.48.90.20.1263 > 138.48.32.99.23: S 1452639370:1452639370(0) win 512 <mss 1460> [tos 0x10]
18:12:55.789794 138.48.32.99.23 > 138.48.90.20.1263: S 3266034444:3266034444(0) ack 1452639371 win 8760 <mss 1460> DF)
18:13:13.073673 138.48.90.1.520 > 255.255.255.255.520: rip-resp 20: 138.48.48.0(1) 138.48.42.0(2)[|rip]
18:13:13.373266 138.48.32.99.23 > 138.48.90.20.1263: F 282:282(0) ack 88 win 8760 (DF)
18:13:13.373856 138.48.90.20.1263 > 138.48.32.99.23: F 88:88(0) ack 283 win 31744 [tos 0x10]
```

Fichier de sortie de TCPdump pour une connexion telnet.

On voit clairement le paquet de demande d'ouverture (1ère ligne), le paquet d'accusé de réception de cette demande (2ème ligne), le paquet de demande de fermeture (4ème ligne), le paquet d'accusé de réception de cette demande (5ème ligne). La 3ème ligne est un paquet dû au protocole RIP. On voit également qu'aucun paquet de données n'a été repris.

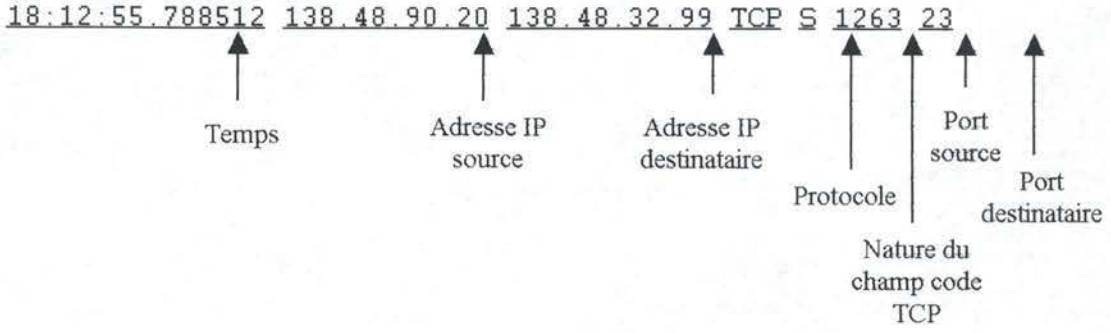
Cependant, cet exemple, bien que de petite taille, montre clairement que la complexité et la variété des termes de la syntaxe rendraient très difficile (mais pas impossible) à réaliser un adaptateur de format permettant de créer un fichier en format NADF à partir de ce genre de fichier de sortie. Pour nous simplifier le problème et pour une question de facilité, nous allons utiliser un petit programme rédigé en *awk*[3] qui va nous permettre de mettre les fichiers de sortie de TCPdump sous une forme beaucoup plus normalisée[4]. L'application de ce petit convertisseur transforme le fichier de sortie TCPdump de l'exemple précédant en un fichier du format suivant :

```
18:12:55.788512 138.48.90.20 138.48.32.99 TCP S 1263 23
18:12:55.789794 138.48.32.99 138.48.90.20 TCP S 23 1263
18:13:13.373266 138.48.32.99 138.48.90.20 TCP F 23 1263
18:13:13.373856 138.48.90.20 138.48.32.99 TCP F 1263 23
```

Fichier de sortie de TCPdump après normalisation par le programme awk..

[2] Ce sera d'autant plus le cas ici puisque nous comptons l'utiliser à la frontière entre le réseau des facultés et Internet, c'est à dire à l'endroit où les conversations seront le plus entrelacées.
[3] Ce n'est pas le seul langage qui aurait pu convenir.
[4] Voir annexes

On obtient alors quelque chose de beaucoup plus simple à adapter en NADF. En effet, nous avons :

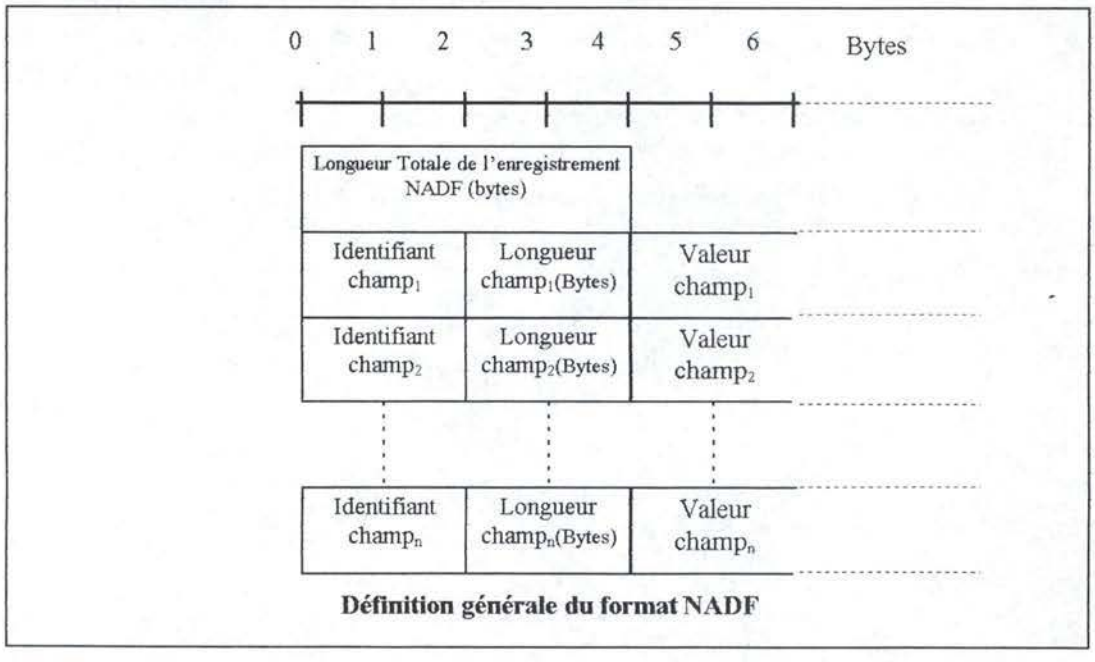


On peut remarquer que cet enregistrement ne contient que 7 champs, on n'a pas pris compte des champs contenant les numéros de séquence de TCP (*sequence number*). Dans la pratique, il nous sera donc impossible de détecter avec précision une attaque comme SEQUENCE NUMBER ATTACK (pages 107 et 108)[5]. En fait, il s'agit ici d'étudier la faisabilité de la conception d'un outil de sécurité à l'aide d'ASAX. Cependant, si dans l'avenir, on désire détecter ce genre d'attaque, les modifications à apporter sont simples.

Dans l'enregistrement ci-dessus, on élimine également les informations parasites que constituent les paquets induits par le protocole RIP. Cela contribue encore à faire diminuer le nombre de paquets à traiter par unité de temps.

3.3 Le format NADF.

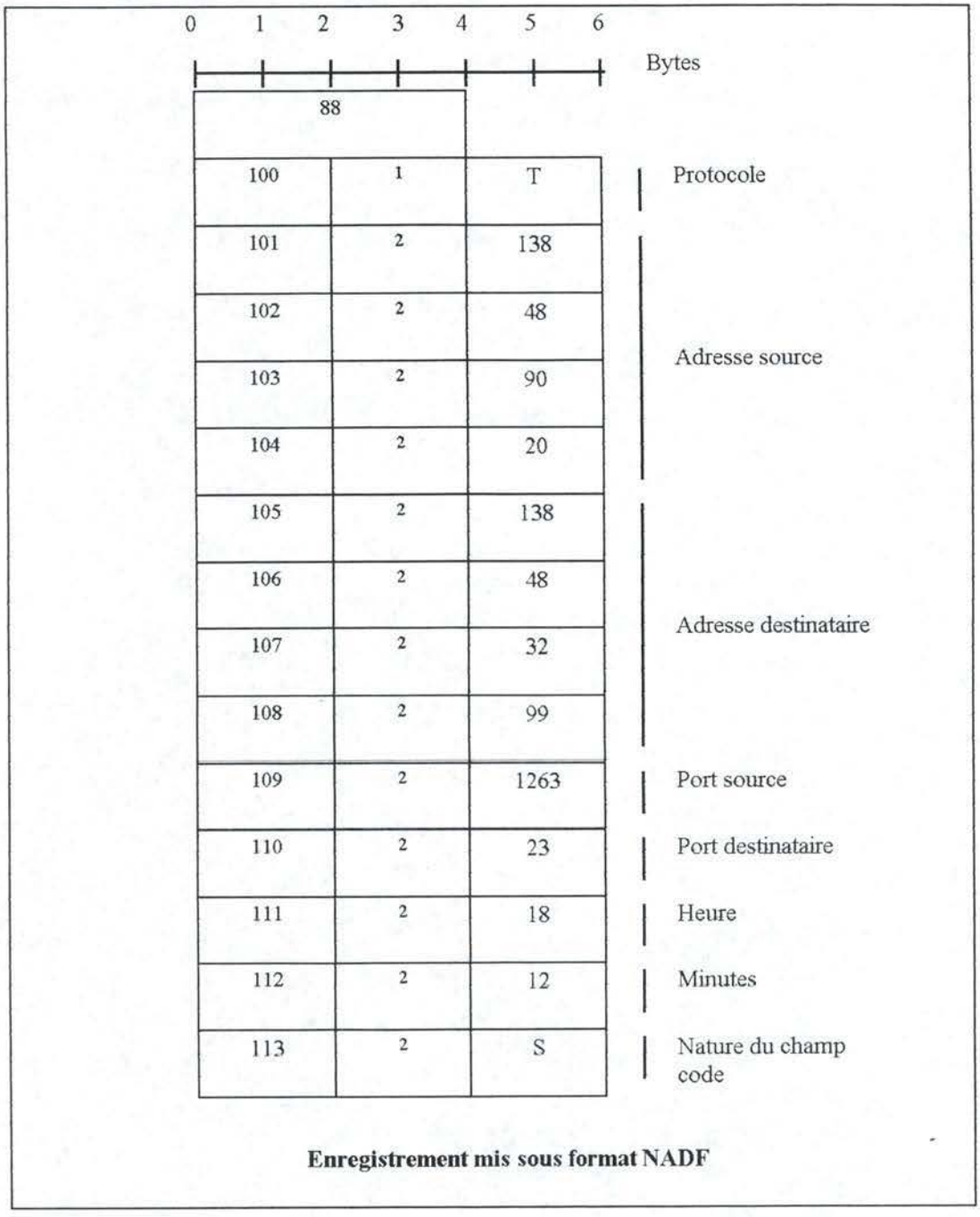
Si on reprend la définition du format NADF présentée dans la deuxième partie de ce mémoire, on doit obtenir le format suivant :



Définition générale du format NADF

[5] Nous allons voir cependant qu'il est possible de contourner partiellement le problème.

Ce qui, pour l'exemple ci-dessus va se matérialiser en :



Enregistrement mis sous format NADF

Conventions :

- Le protocole TCP est symbolisé par la lettre 'T', UDP par la lettre 'U'.

- La nature du champ code sera respectivement notée 'S', 'F', et 'A' pour SYN, FIN et ACK.

Remarque : les enregistrements NADF peuvent avoir des longueurs variables selon que l'on travaille avec UDP ou TCP.
En effet, avec UDP, le champ CODE n'existe pas. De plus, avec TCP on pourra avoir 1 ou 2 valeurs pour CODE. Les différentes longueurs que l'on obtient sont donc de 88 (UDP), 92 (TCP et 1 CODE) et 96 (TCP et 2 CODE).

L'enregistrement fourni par TCPdump doit donc subir les conversions suivantes :

```
:12:55.788512 138.48.90.20.1263 > 138.48.32.99.23: S 1452639370:1452639370(0) win 512 <mss 1460> [tos 0x10]
```

Enregistrement issu de TCPdump

Transformation par programme awk

```
18:12:55.788512 138.48.90.20 138.48.32.99 TCP S 1263 23
```

Adaptateur de format

Format NADF

88		
100	1	T
101	2	138
102	2	48
103	2	90
104	2	20
105	2	138
106	2	48
107	2	32
108	2	99
109	2	1263
110	2	23
111	2	18
112	2	12
113	2	S

3.4 Fichier de description de données (*data description file*).

De ce qui précède, on déduit le fichier de description des données (*data description file*) suivant :

```
A tcpdump
B Unix_system_V
C 24-02-1997
D dos-edit
E MS_DOS version 6.0
F 27-02-1997

1 100
2 char
3 char
4 PROTOCOL
5 Type de protocole utilis,

1 101
2 unsigned_int
3 unsigned_int
4 SRC_ADDR_PART_1
5 Premier quart de l'adresse IP source

1 102
2 unsigned_int
3 unsigned_int
4 SRC_ADDR_PART_2
5 Deuxième quart de l'adresse IP source

1 103
2 unsigned_int
3 unsigned_int
4 SRC_ADDR_PART_3
5 Troisième quart de l'adresse IP source

1 104
2 unsigned_int
3 unsigned_int
4 SRC_ADDR_PART_4
5 Quatrième quart de l'adresse IP source

1 105
2 unsigned_int
3 unsigned_int
4 DEST_ADDR_PART_1
5 Premier quart de l'adresse IP de destination

1 106
2 unsigned_int
3 unsigned_int
4 DEST_ADDR_PART_2
5 Deuxième quart de l'adresse IP de destination

1 107
2 unsigned_int
3 unsigned_int
4 DEST_ADDR_PART_3
5 Troisième quart de l'adresse IP de destination

1 108
2 unsigned_int
3 unsigned_int
4 DEST_ADDR_PART_4
5 Quatrième quart de l'adresse IP de destination

1 109
2 unsigned_int
3 unsigned_int
4 SRC_PORT
5 Numéro de port source

1 110
2 unsigned_int
3 unsigned_int
4 DEST_PORT
5 Numéro de port de destination

1 111
2 unsigned_int
3 unsigned_int
4 HOUR
5 Heure

1 112
2 unsigned_int
3 unsigned_int
4 MIN
5 Minute

1 113
2 char
3 char
4 FIRST_CODE
5 Code TCP présent

1 114
2 char
3 char
4 SECOND_CODE
5 Code TCP présent
```

Fichier de description de données (*Data Description File*).

L'adaptateur de format va être décrit dans ce qui suit. Un adaptateur de format peut être réalisé sous une forme *on-line* ou *off-line*. La version *off-line* est très utile en ce qui concerne la mise au point des différentes règles RUSSEL destinées à la détection des différents schémas d'attaque. En effet, on peut créer soi-même les enregistrements et ainsi simuler les attaques. La version *on-line* s'avère cependant indispensable à réaliser si l'on veut concevoir un outil de protection qui permette une **détection immédiate** des attaques.

3.5 Adaptateur *off-line*.

Le programme de l'adaptateur *off-line* se trouve en annexes. La programmation de cet adaptateur n'a pas poser de véritable problème. La seule chose importante à savoir, c'est qu'un entier (*int*) en C est codé sur 2 bytes sous UNIX et qu'un entier long (*long int*) est codé sur 4 bytes. Remarquons également qu'un caractère ne prend qu'un byte mais que le format NADF veut qu'une place d'un nombre paire de byte lui soit réservée→2bytes.

3.6 Adaptateur *on-line*.

La programmation de l'adaptateur *on-line* ne présente pas plus de difficultés que celui *off-line*. Pour celui-ci, il faut rédiger trois routines : *vopen*, *vread*,*vclose*.

Une manière simple de programmer l'adaptateur *on-line* : transformer l'adaptateur *off-line* pour qu'il fonctionne avec des procédures de nom *vopen*, *vread*, *vclose*. Le programme de l'adaptateur *on-line* et les description des procédures *vread*, *vopen* et *vclose* se trouvent en annexes.

4 Deux exemples de conception de règle de détection en RUSSEL

4.1 Introduction

Dans ce qui suit, deux exemples de conception de requêtes RUSSEL vont être détaillés. Le premier exemple porte sur un comportement suspect dont l'administrateur du réseau désirerait être averti :$TELNET_{entrant}$/$TELNET_{sortant}$. Le deuxième exemple porte sur une attaque présentée dans la deuxième partie de ce mémoire :SEQUENCE NUMBER ATTACK.

4.2 SEQUENCE NUMBER ATTACK[6]

4.2.1 Introduction

Comme nous l'avons signalé au point 3.2 (page 157) les numéro de séquence ne sont pas repris dans les champs du fichier NADF. Cependant, il existe un moyen de contourner cette difficulté pour l'attaque qui va suivre. Cette attaque contient différents aspects d'autres attaques et il serait intéressant de montrer qu'il est relativement simple d'implémenter un système de détection avec RUSSEL, même pour un cas assez complexe comme celui-ci.

Remarque : Si nous voulions nous intéresser aux numéros de séquence pour pouvoir rédiger des règles de détection, il faudrait donc apporter quelques changements aux champs précédemment utilisés. Il faudrait en effet ajouter les numéros de séquence TCP aux différents champs que l'on avait juste avant transformation en format NADF. Il nous faudrait également modifier le fichier de description de données en conséquence en ajoutant deux champs : SEQ_NUM_SYN et SEQ_NUM_ACK par exemple. Les modifications qu'il faudrait apporter aux deux versions de l'adaptateur de format seraient également très simples à réaliser[7].

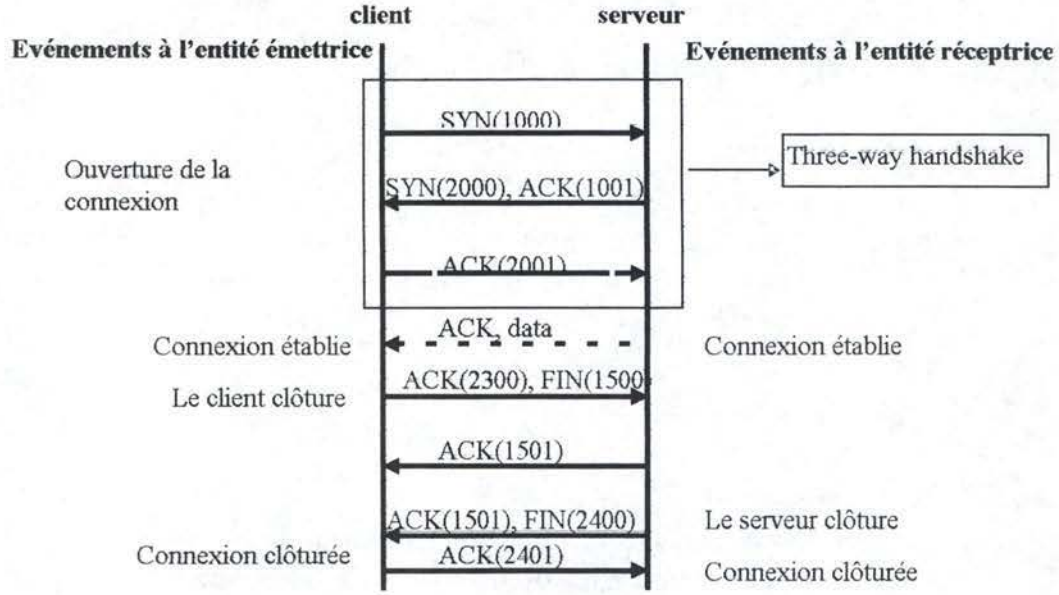
4.2.2 Bref rappel des principes de cette attaque :

Le protocole de transport (TCP) fournit des circuits virtuels sûrs (*reliable*) aux processus utilisateurs. Les paquets perdus ou endommagés sont retransmis; les paquets entrant sont réordonnés si nécessaire. L'ordre est maintenu grâce au numéro de séquence (*sequence number*) contenu chaque paquet. Chaque *byte* de données envoyé, tout comme chaque paquet d'ouverture ou de fermeture de communication, est numéroté individuellement.

Un numéro de séquence est codé sur 32 bits, il est donc compris entre 0 et 4 294 967 295. Dans une même machine, le numéro de séquence est incrémenté de 128 000 par seconde et 64 000 à chaque connexion.

[6] Pour les détails de cette attaque, il faut se reporter à la deuxième partie de ce mémoire : SEQUENCE NUMBER ATTACK, page 107-108.

[7] Notons que la simplicité avec laquelle on peut modifier l'adaptateur de format joue en faveur d'ASAX, le rendant très pratique à utiliser et facile à adapter aux besoins.



Cette figure représente un exemple d'une session TCP. Le paquet initial, avec le bit SYN (SYNchronize ou requête d'ouverture) activé, transmet le numéro de séquence initial pour son côté de la connexion. Le numéro de séquence initial est aléatoire. Tous les paquets suivants auront le bit ACK (ACKnowledge, ou accusé de réception) activé.

Tous les paquets envoyés, exception faite du tout premier, contiennent un numéro d'accusé de réception *(acknowledgment)*; c'est le numéro de séquence des derniers *bytes* reçus augmenté de 1.

Chaque paquet TCP contient, entre autres, quatre champs particuliers (appelé aussi *socket*) :

< adresse de **l'hôte local** ; numéro de port de **l'hôte local** ;
adresse de l'hôte **destinataire** ; numéro de port **destinataire** >

Les numéros de séquence mentionnés plus haut ont une autre fonction. Parce que le numéro de séquence change constamment pour chaque nouvelle connexion, il est possible pour TCP de détecter de vieux paquets d'occurrences précédentes du même circuit (p.e. : utilisation précédante du même quadruplet identifiant). Une connexion ne peut donc être complètement établie tant que chacune des deux parties n'a pas envoyé un accusé de réception du numéro de séquence initial à l'autre.
ER.

Si un pirate peut prédire la séquence initiale de sa cible, alors il est possible, pour l'agresseur, de faire croire à la cible qu'elle communique avec une machine de confiance (*trusted*). Dans ce cas, les protocoles qui reposent sur l'adresse IP pour l'authentification (p.e. : Rlogin) peuvent être exploités pour pénétrer le système cible.

La séquence normale d'établissement d'une connexion TCP comprend une Three-way Handshake (triple poignée de mains). Le client (C) choisit et transmet son numéro de séquence initial ISNc, le serveur (S) accuse réception et envoie son propre

numéro de séquence ISNs. Le client, à son tour, accuse réception de ce paquet. Après cela la connexion est établie et les données peuvent être envoyées.

C →S : SYN(ISNc)
S →C : SYN(ISNs), ACK(ISNc+1)
C→ S : ACK(ISNs+1)
C →S : données et/ou
S →C : données

Ceci signifie que C doit d'abord avoir reçu ISNs avant que la conversation puisse débuter.

Supposons qu'il soit possible pour un intrus X de prédire ISNs. Dans ce cas, X pourrait envoyer la séquence suivante afin de se faire passer pour un hôte de confiance T (trusted) :

X →S : SYN(ISNx),(X prétend que SRC=T)
S →T : SYN(ISNs), ACK(ISNx+1)
X→ S : ACK(ISNs+1), (X prétend que SRC=T)
X →S : ACK(ISNs+1), (X prétend que SRC=T), mauvaises données

1°. Bien que le message S→T n'arrive jamais à X, X est cependant capable de deviner son contenu et peut ainsi envoyer des données[8]. Pour deviner le numéro de séquence, le pirate peut soit « écouter » les connexions TCP issues de la cible (par un des moyens présentés dans la deuxième partie), soit se connecter légitimement à la cible juste avant de mener son attaque (s'il en a la possibilité). Par l'analyse des numéros de séquence contenus dans les paquets il peut se faire une idée de l'ordre de grandeur des numéros de séquence qu'il va devoir employer pour mener son attaque[9]. Ne pouvant deviner avec exactitude le numéro de séquence, il va devoir effectuer plusieurs tentatives. Certaines tentatives échouant se caractérisant par un **Reset** (RST) de la connexion. C'est grâce à cela que l'on va pouvoir détecter cette attaque.

2°. Le message S→T : SYN(ISNs), ACK(ISNx) ne disparaît pas comme par enchantement ; en fait, si le véritable T reçoit ce message, il va essayer de réinitialiser la connexion. Pour éviter cela, il suffit de submerger le port serveur de T avec des demandes de connexions (SYN). Cela va générer un *overflow* qui va faire en sorte que le message S→T va être perdu[10]. En effet, chaque machine possède une *Backlock Queue* qui est en fait le nombre maximum de connexions TCP qu'elle peut admettre pour un même *socket*. Ce nombre est typiquement de 5 ou 6. Une fois ce nombre atteint, toute requête adressée sera automatiquement rejetée. Cette attaque est une attaque de **rejet de service** que l'on peut également détecter facilement.

[8] Remarquons que lors de cette attaque, le pirate est aveugle.
[9] Dans les systèmes Berkeley, le numéro de séquence initial est incrémenté qu'une même constante une fois par seconde, et par la moitié de cette constante à chaque ouverture de connexion. Donc, si on observe l'ISNs utilisé lors d'une connexion légitime, on pourra calculer, avec un grand degré de sûreté, l'ISNs' qui sera utilisé pour la prochaine tentative de connexion.
[10] On peut également attendre que la machine T soit déconnectée ou rebootée.

3°. Cette attaque n'est, comme nous l'avons dit, efficace que pour les applications dont l'authentification repose sur l'adresse IP : par exemple rsh, rlogin. Dans un réseau local, il est normal, voire important, qu'un hôte ne puisse être *trusted* que s'il fait partie lui-même de ce réseau local[11]. Un pirate, externe au réseau des Facultés, devrait donc pratiquer un attaque d'IP-Spoofing (voir deuxième) pour se faire passer pour un hôte interne. Il suffirait donc de détecter, au niveau du *sniffeur*, une adresse du type 138.48.x.y venant de l'extérieur pour détecter ce genre d'attaque. Nous montrerons plus loin comment rédiger une règle RUSSEL permettant de le faire, mais notons cependant que ce genre de comportement est déjà rendu impossible au niveau de Belnet. En effet, ceux-ci nous garantissent qu'aucun paquet d'adresse source 138.48.x.y ne peut provenir de l'extérieur du réseau des Facultés.

Les trois sous-attaques ci-dessus font l'objet de rédaction de règles en RUSSEL :

4.2.3 Ecriture des règles de détection en langage RUSSEL

4.2.3.1 Détection de RESET

Pour détecter cette attaque de cette manière, il faut établir des statistiques sur le nombre de RESET survenant habituellement et en absence d'attaque sur un *socket*. Si ce nombre n'est pas trop élevé, il est alors possible de détecter une tentative d'attaque en se fixant une limite du nombre de RESET/seconde que l'on estime être normal. Tout dépassement de cette limite (ici, 2 RESET en moins d'une minute) pouvant être un signe d'attaque. Cela donne[12] :

```
global internal nbr : integer;

init_action;
begin
        nbr:=0;
        trigger off for_next reset_1
end;

rule reset_1;
begin
        if
                CODE='R' ;
                -->     hr :=HOUR ;
                        mn :=MIN ;
                        trigger_off for_next reset_2(nbr, hr, min, SRC_ADDR_PART_1,
SRC_ADDR_PART_2, SRC_ADDR_PART_3, SRC_ADDR_PART_4, DEST_ADDR_PART_1,
DEST_ADDR_PART_2, DEST_ADDR_PART_3, DEST_ADDR_PART_4, SRC_PORT,
DEST_PORT)
        fi;
        trigger off for_next reset_1
end;
```

Avec cette règle on détecte la première apparition d'un paquet contenant RST et on passe le temps initial et les caractéristiques identifiant le paquet comme paramètres

[11] C'est à dire que le fichier *.rhosts* ne devrait contenir que des adresses du réseau local.

[12] Ces programmes ne sont pas tout à fait syntaxiquement corrects pour une raison de plus grande lisibilité.

```
rule reset_2( hr, mn, src_part1, src_part2, src_part3, src_part4, dest_part1, dest_part2, dest_part3,
dest_part4, s_port, d_port : integer);
begin
        if
                        SRC_PORT=s_port
                and     DEST_PORT=d_port
                and     SRC_ADDR_PART_1=src_part1
                and     SRC_ADDR_PART_2= src_part2
                and     SRC_ADDR_PART_3= src_part3
                and     SRC_ADDR_PART_4= src_part4
                and     DEST_ADDR_PART_1=dest_part1
                and     DEST_ADDR_PART_2= dest_part2
                and     DEST_ADDR_PART_3= dest_part3
                and     DEST_ADDR_PART_4= dest_part4
                and     CODE='R'
                and     (HOUR*60)+MIN<(hr*60+mn)+1 ;
                →       nbr :=nbr+1 ;
                        trigger off for_next warning(nbr, hr, min, SRC_ADDR_PART_1,
SRC_ADDR_PART_2, SRC_ADDR_PART_3, SRC_ADDR_PART_4, DEST_ADDR_PART_1,
DEST_ADDR_PART_2, DEST_ADDR_PART_3, DEST_ADDR_PART_4, SRC_PORT,
DEST_PORT)
        fi;
        trigger off for_next reset_2(nbr, hr, min,  SRC_ADDR_PART_1, SRC_ADDR_PART_2,
SRC_ADDR_PART_3, SRC_ADDR_PART_4, DEST_ADDR_PART_1, DEST_ADDR_PART_2,
DEST_ADDR_PART_3, DEST_ADDR_PART_4, SRC_PORT, DEST_PORT)
end;
```

Teste, sur base des caractéristiques du paquet, si celui-ci est un second RST et regarde si moins d'une minute s'est écoulée. Si oui, on déclenche un message d'avertissement.

```
rule warning (num , hr, mn, src_part1, src_part2, src_part3, src_part4, dest_part1, dest_part2,
dest_part3, dest_part4, s_port, d_port : integer);
var fd,code:integer;
begin
        println('************************************');
        println('SEQUENCE NUMBER ATTACK supposée.');
        println('Informations dans le fichier SEQNUM',nbr,'.atk');
        println('************************************');
        fd:=createNADF('SEQNUM',num,'.atk');
        if
                fd<0 --> println('cannot create SEQNUM',num,'.atk');
                true --> begin

                                if
                                        writeNADF(fd) <0 → println('write error')
                                fi ;
                                println('------------------------------------------------------------') ;
                                println('Time :',hr,'.',mn) ;
                                println('from :', src_part1, '.',src_part2, '.', src_part3, '.',
src_part4, '--',s_port) ;
                                println('to :', dest_part1, '.',dest_part2, '.', dest_part3, '.',
dest_part4, '--',dest_port) ;
                                println('------------------------------------------------------------')
                        end
        fi ;
        code := closeNADF(fd) ;
        if fd<0 --> println('closing error :',code) fi
end.
```

On affiche un message d'avertissement à l'écran et on place les détails dans un fichier SEQNUMx.atk

Dans **rule reset_1** on est à l'affût d'un paquet RST. Si un tel paquet survient, on déclenche **rule reset_2** dans laquelle on va tester si un paquet RST avec la même origine et la même destination (passage de paramètres) que dans **rule reset_1** peut être détecter endéans la minute.

4.2.3.2 Détection de rejet de service

Le principe de détection de cette attaque est fort proche du précédant. En effet, si on détecte plus de 10 paquets SYN en moins d'une minute pour une même paire de *socket* (alors que le basklosk queue est de 5 ou 6) il convient d'en être averti.

```
global internal nbr,cpt : integer;

init_action;
begin
        nbr:=0;
        cpt :=10 ;
        trigger off for_next reset_1
end;

rule denial_1;
begin
        if
                CODE='S' ;
                -->     hr :=HOUR ;
                        mn :=MIN ;
                        cpt :=cpt-1 ;
                        trigger_off for_next denial_2(cpt, nbr, hr, min,  SRC_ADDR_PART_1,
SRC_ADDR_PART_2, SRC_ADDR_PART_3, SRC_ADDR_PART_4, DEST_ADDR_PART_1,
DEST_ADDR_PART_2, DEST_ADDR_PART_3, DEST_ADDR_PART_4, SRC_PORT,
DEST_PORT)
        fi;
        trigger off for_next denial_1
end;

rule denial_2(cpt, hr, mn, src_part1, src_part2, src_part3, src_part4, dest_part1, dest_part2, dest_part3,
dest_part4, s_port, d_port : integer);
begin
        if
                        SRC_PORT=s_port
                and     DEST_PORT=d_port
                and     SRC_ADDR_PART_1=src_part1
                and     SRC_ADDR_PART_2= src_part2
                and     SRC_ADDR_PART_3= src_part3
                and     SRC_ADDR_PART_4= src_part4
                and     DEST_ADDR_PART_1=dest_part1
                and     DEST_ADDR_PART_2= dest_part2
                and     DEST_ADDR_PART_3= dest_part3
                and     DEST_ADDR_PART_4= dest_part4
                and     CODE='S'
                and     cpt>0
                →       cpt :=cpt-1 ;
```

Avec cette règle on détecte la première apparition d'un paquet contenant RST et on passe le temps initial, le nombre de paquets/minute et les caractéristiques identifiant le paquet comme paramètres

Teste, sur base des caractéristiques du paquet, si celui-ci est un autre SYN, regarde si moins d'une minute s'est écoulée et si cpt>0. Si oui, on redéclenche cette règle.

```
                    trigger off for_next denial_2(cpt, nbr, hr, min,  SRC_ADDR_PART_1,
SRC_ADDR_PART_2, SRC_ADDR_PART_3, SRC_ADDR_PART_4, DEST_ADDR_PART_1,
DEST_ADDR_PART_2, DEST_ADDR_PART_3, DEST_ADDR_PART_4, SRC_PORT,
DEST_PORT)
        fi;
        fi cpt > 0 → trigger off for_next denial_2(cpt, nbr, hr, min,  SRC_ADDR_PART_1,
SRC_ADDR_PART_2, SRC_ADDR_PART_3, SRC_ADDR_PART_4, DEST_ADDR_PART_1,
DEST_ADDR_PART_2, DEST_ADDR_PART_3, DEST_ADDR_PART_4, SRC_PORT,
DEST_PORT)
        fi;
        if cpt=0 →trigger off for_next warning(nbr, hr, min, SRC_ADDR_PART_1,
SRC_ADDR_PART_2, SRC_ADDR_PART_3, SRC_ADDR_PART_4, DEST_ADDR_PART_1,
DEST_ADDR_PART_2, DEST_ADDR_PART_3, DEST_ADDR_PART_4, SRC_PORT,
DEST_PORT)
        fi
end;

rule warning (num , hr, mn, src_part1, src_part2, src_part3, src_part4, dest_part1, dest_part2,
dest_part3, dest_part4, s_port, d_port : integer);
var fd,code:integer;
begin
        println('************************************');
        println('DENIAL OF SERVICE ATTACK supposée.');
        println('Informations dans le fichier DENSERV',nbr,'.atk');
        println('************************************');
        fd:=createNADF('SEQNUM',num,'.atk');
        if
                fd<0 --> println('cannot create SEQNUM',num,'.atk');
                true --> begin

                                if
                                        writeNADF(fd) <0 → println('write error')
                                fi ;
                                println('------------------------------------------------------------') ;
                                println('Time :',hr,'.',mn) ;
                                println('from :', src_part1, '.',src_part2, '.', src_part3, '.',
src_part4, '--',s_port) ;
                                println('to :', dest_part1, '.',dest_part2, '.', dest_part3, '.',
dest_part4, '--',dest_port) ;
                                println('------------------------------------------------------------')
                        end
        fi ;
        code := closeNADF(fd) ;
        if fd<0 --> println('closing error :',code) fi
end.
```

On affiche un message d'avertissement à l'écran et on place les détails dans un fichier DENSERVx.atk

4.2.3.3 Détection d'IP-spoofing

Ici, il s'agit de détecter un paquet provenant de l'extérieur du réseau et ayant une adresse source 138.48.x.y (et évidemment une adresse destinataire 138.48.x'.y'). La règle est simple à écrire, la seule différence est que l'écriture du message se fait dans un fichier `spoof.out`.

```
global external count : integer;

init_action;
var fd:integer;
begin
        fd:=createNADF('spoof.out');
        if
                fd<0 --> println('cannot create spoof.out');
                true --> begin
                                trigger off for_next ip_spoof(fd);
                                trigger off at_completion echo_result2(fd);
                        end
        fi
end;

rule ip_spoof(fd:integer);
begin
        if
                SRC_ADDR_PART_1=DEST_ADDR_PART_1
                and SRC_ADDR_PART_2=DEST_ADDR_PART_2
                -->begin
                        if
                                writeNADF(fd) < 0 --> println('write error')
                        fi;
                        println('Ip_spoofing from
:',SRC_ADDR_PART_1,'.',SRC_ADDR_PART_2,'.',SRC_ADDR_PART_3,'.',SRC_ADDR_PART_4
);
                        println('          to
:',DEST_ADDR_PART_1,'.',DEST_ADDR_PART_2,'.',DEST_ADDR_PART_3,'.',DEST_ADDR_PA
RT_4);
                        count:=count+1;
                end
        fi;
        trigger off for_next ip_spoof;


rule echo_result2(fd:integer);
var code:integer;
begin
        println('');
        code:=closeNADF(fd);
        if code <0 --> println('Close error', code) fi;
        println('');
        println('----------------------------------------');
        println(count,'tentative(s) d'HOST-SPOOFING');
        println('----------------------------------------');
end.
```

4.3 $TELNET_{entrant}/TELNET_{sortant}$

Ici, il s'agit de détecter un *telnet*[13] entrant (d'origine externe au réseau des facultés et à destination de celles-ci), suivi d'un *telnet* sortant (issu des Facultés). Ce genre de comportement est fréquent chez les pirates. Cette manœuvre leur permet en effet de brouiller les pistes lorsqu'ils attaque un autre site. Telnet repose sur le protocole de transport TCP, cette règle va donc faire intervenir des paquets d'ouverture et de fermeture de connexions TCP.

```
init_action;
        trigger off for_next TCP_open_part_1;

rule TCP_open_part_1;
begin
        if
                DEST_PORT=23
                and DEST_ADDR_PART_1=138
                and DEST_ADDR_PART_2=48
                and FIRST_CODE='S'
                and not present SECOND_CODE
                -->
                        begin
                                trigger off for_next TCP_open_part_2(HOUR, MIN,
SRC_ADDR_PART_1, SRC_ADDR_PART_2, SRC_ADDR_PART_3, SRC_ADDR_PART_4,
DEST_ADDR_PART_1, DEST_ADDR_PART_2, DEST_ADDR_PART_3, DEST_ADDR_PART_4,
SRC_PORT, DEST_PORT);
                        end;
        fi;
        trigger off for_next TCP_open_part_1;
end;

rule TCP_open_part_2(hr, mn, src_part1, src_part2, src_part3, src_part4, dest_part1, dest_part2,
dest_part3, dest_part4, s_port, d_port : integer);
var client_port : integer;
begin
        if
                SRC_PORT=23
                and     DEST_PORT=s_port ;
                and     SRC_ADDR_PART_1=dest_part1
                and     SRC_ADDR_PART_2=dest_part2
                and     SRC_ADDR_PART_3=dest_part3
                and     SRC_ADDR_PART_4=dest_part4
                and     DEST_ADDR_PART_1=src_part1
                and     DEST_ADDR_PART_2=src_part2
                and     DEST_ADDR_PART_3=src_part3
                and     DEST_ADDR_PART_4=src_part4
                and     FIRST_CODE=S
                and not present SECOND_CODE
                -->
                        begin
```

Détection du premier paquet (SYN) de demande d'ouverture d'un TELNET vers une station des facultés

Détection de la réponse (ACK) et envoi du (SYN)

[13] Numéro de port TCP : 23.

```
                client_port:=DEST_PORT;
                trigger off for_next TCP_NOT_CLOSED(client_port, hr,mn,
src_part1, src_part2, src_part3, src_part4, dest_part1, dest_part2, dest_part3, dest_part4, s_port,
d_port);
            end;
        fi;
        trigger off for_next TCP_open_part_2(hr, mn, src_part1, src_part2, src_part3, src_part4,
dest_part1, dest_part2, dest_part3, dest_part4, s_port, d_port);
end;


rule TCP_NOT _CLOSED(client_port, hr,mn, src_part1, src_part2, src_part3, src_part4, dest_part1,
dest_part2, dest_part3, dest_part4, s_port, d_port : integer);
begin
        if                                          Détection de la non fermeture de la
            not (                                   connexion
                (
                    (       SRC_PORT=23
                    and     DEST_PORT=client_port
                    and     SRC_ADDR_PART_1=dest_part1
                    and     SRC_ADDR_PART_2=dest_part2
                    and     SRC_ADDR_PART_3=dest_part3
                    and     SRC_ADDR_PART_4=dest_part4
                    and     DEST_ADDR_PART_1=src_part1
                    and     DEST_ADDR_PART_2=src_part2
                    and     DEST_ADDR_PART_3=src_part3
                    and     DEST_ADDR_PART_4=src_part4
                    )
                or
                    (       SRC_PORT=client_port
                    and     DEST_PORT=23
                    and     SRC_ADDR_PART_1=src_part1
                    and     SRC_ADDR_PART_2=src_part2
                    and     SRC_ADDR_PART_3=src_part3
                    and     SRC_ADDR_PART_4=src_part4
                    and     DEST_ADDR_PART_1=dest_part1
                    and     DEST_ADDR_PART_2=dest_part2
                    and     DEST_ADDR_PART_3=dest_part3
                    and     DEST_ADDR_PART_4=dest_part4
                    )
                )

                and     (FIRST_CODE=F or SECOND_CODE=F)
            )
        -->
                begin
                    trigger off for_next TCP_DETECT(client_port, hr, mn, src_part1,
src_part2, src_part3, src_part4, dest_part1, dest_part2, dest_part3, dest_part4, s_port, d_port);
                end;
        fi;
        trigger off for_next TCP_NOT_CLOSED(client_port, hr,mn, src_part1, src_part2, src_part3,
src_part4, dest_part1, dest_part2, dest_part3, dest_part4, s_port, d_port);

end;
```

```
rule TCP_DETECT(client_port, hr, mn, src_part1, src_part2, src_part3, src_part4, dest_part1,
dest_part2, dest_part3, dest_part4, s_port, d_port : integer);
var time1,time2 : integer;
begin
        time1:=(hr*60)+mn;
        time2:=(HOUR*60)+MIN;
        if
                time2-time1<15
                and     SRC_ADDR_PART_1=dest_part1
                and     SRC_ADDR_PART_2=dest_part2
                and     SRC_ADDR_PART_3=dest_part3
                and     SRC_ADDR_PART_4=dest_part4
                and     DEST_PORT=23
                -->
                        begin
                                nbr :=nbr+1 ;
                                trigger off for_next warning(nbr, hr, mn, src_part1, src_part2,
src_part3, src_part4, dest_part1, dest_part2, dest_part3, dest_part4, s_port, d_port)
                        end
        fi;
end;
```

Détection du paquet de demande d'ouverture vers une station extérieure en moins de 15 minutes

```
rule warning (num , hr, mn, src_part1, src_part2, src_part3, src_part4, dest_part1, dest_part2,
dest_part3, dest_part4, s_port, d_port : integer);
var fd,code:integer;
begin
        println('*************************************');
        println('TELNETentrant/sortant en moins de 15 minute détecté.');
        println('Informations dans le fichier TELin/out',nbr,'.cprt');
        println('*************************************');
        fd:=createNADF('TELin/out',num,'.cprt');
        if
                fd<0 --> println('cannot create SEQNUM',num,'.atk');
                true --> begin

                                if
                                        writeNADF(fd) <0 → println('write error')
                                fi ;
                                println('-----------------------------------------------------------') ;
                                println('Time :',hr,'.',mn) ;
                                println('from :', src_part1, '.',src_part2, '.', src_part3, '.',
src_part4, '--',s_port) ;
                                println('to :', dest_part1, '.',dest_part2, '.', dest_part3, '.',
dest_part4, '--',dest_port) ;
                                println('-----------------------------------------------------------')
                        end
        fi ;
        code := closeNADF(fd) ;
        if fd<0 --> println('closing error :',code) fi
end.
```

On affiche un message d'avertissement à l'écran et on place les détails dans un fichier Tlin/outx.cprt

Conclusion

L'objectif de ce mémoire était d'étudier la possibilité d'utiliser ASAX pour la conception d'un outil de sécurité (surveillance) des réseaux.

Pour y arriver, il a fallu tout d'abord s'intéresser au fonctionnement détaillé de différents protocoles (notamment TCP/IP et Ethernet) afin de pouvoir bien comprendre les attaques.

Ensuite, après avoir vu ce qu'il était possible de faire en matière d'attaques, ce sont les solutions qui ont été abordées, et plus précisément la possibilité de concevoir un outil de surveillance des réseaux à l'aide d'ASAX en configuration firewall. L'objectif final étant de vérifier la validité de l'outil réalisé en configuration réelle, sur des machines et un réseau réel, à savoir, le réseau des Facultés.

Il a ainsi été démontré qu'il est possible d'utiliser ASAX pour concevoir un outil de surveillance (et donc de sécurité) pour les réseaux informatiques.

Perspectives :

Bien que nous nous soyons attardés longuement sur les attaques, bien peu d'entre elles ont finalement été implémentées puisque nous nous sommes principalement concentré sur la faisabilité de la réalisation d'un outil de sécurité. Cependant, il serait peut-être intéressant de rédiger des règles RUSSEL permettant de détecter ces attaques. On pourrait tenter de le faire dans une optique de réalisation d'un outil de sécurité plus complet. En effet, les besoins en sécurité demandés par les facultés ne sont pas ceux d'autres organisations. Comme nous l'avons expliqué, ASAX est un outil très flexible qui peut s'adapter facilement à n'importe quel outil de monitoring de réseau et il est relativement aisé de programmer un adaptateur de format.

Pour un administrateur de sécurité il serait également pratique de disposer d'un outil possédant une interface lui permettant de cerner plus rapidement un problème ou d'introduire de nouvelles configurations le plus facilement possible. ASAX étant programmé en langage C, il serait tout à fait possible de concevoir une telle interface avec les outils disponibles maintenant sur le marché (POWER BUILDER,etc.).

Comme nous l'avons vu dans la deuxième partie, un firewall fonctionnant sur le principe du filtrage de paquets est particulièrement délicat à configurer. En effet, l'inversion de lignes, dans la programmation de la liste d'accès, peut rendre le filtrage tout fait inefficace (du moins il ne fait pas ce qu'on voulait qu'il fasse réellement). On pourrait, dès lors, utiliser ASAX et son système de règle pour vérifier la bonne configuration du firewall en vérifiant qu'aucun paquet indésirable n'est autorisé à passer.

Annexes

1 Ports TCP, ports UDP et description

Voici les mesure à prendre pour configurer son filtre d'après Cheswick et Bellovin :

Port	Protocol	Name	Description
1	TCP	tcpmux	The TCP port multiplexer. Not very common. Cannot accept some, reject others.
7	UDP, TCP	echo	An echo server; useful for seeing if a machine is alive. A higher level equivalent of ICMP Echo (ping).
9	UDP, TCP	discard	The /dev/null of the Internet. Harmless.
11	TCP	systat	Occasionally (but rarely) connected to netstat, w, or ps. If you do that sort of thing and you shouldn't block this.
13	UDP, TCP	daytime	The time of day, in human-readable form. Harmless.
15	TCP	nestat	See systat.
19	UDP, TCP	chargen	A character stream generator. Some people like reading that sort of thing, and it won't upset your system if they do.
20	TCP	ftp-data	Data channel for FTP. Hard to filter.
21	TCP	ftp	FTP control channel.Allow in only to your FTP server, if any.
23	TCP	telnet	Telnet. Permit only to your login gateway.
25	TCP	smtp	Mail. Allow only to your incoming mail gateways, and make sure those aren't running sendmail.
37	UDP, TCP	time	The time of day, in machine-readable form. Before blocking it (and there's no reason to), remember that ICMP can provide the same data.
43	TCP	whois	Allow in if you run a sanitized whois server; otherwise block.
53	UDP, TCP	domain	Block TCP except from secondary servers.
67	UDP	bootp	Block; it gives out too much information.
69	UDP	tftp	Dangerous but useful. Be careful if you allow it.
70	TCP	gopher	Block.
79	TCP	finger	Allow in only if you run a sanitized finger server, and only to it; block to all other destinations.
83	TCP	http	Also known as WWW. Dangerous but useful. Be careful if you allow it.
87	TCP	link	Rarely used, except by hackers. A lovely port for an alarm.
88	UDP	kerebos	The official Kerberos port. If you allow people to log in to your site, whether directly or via interrealm authentication, you have to open up this port; otherwise, block it. Do the same for 750, the original Kerberos port. Block 749 and 751, the current and original Kerberos password changing ports. The ports used for Kerberos-protected services are probably safe, though.
95	TCP	supdup	Rarely used except by hackers. Another lovely port for an alarm.
109	TCP	pop-2	Unless folks need to read their mail from outside, block it.
110	TCP	pop-3	Ditto.
111	UDP, TCP	sunrpc	Block, but remember that attackers can scan your port number space anyway.
113	TCP	auth	Generally safe. If you block it, don't send an ICMP rejection.
119	TCP	nntp	If you allow it in, use source and destination address filters.
123	UDP	ntp	Safe if you use NTP's own access controls.
144	TCP	NeWS	A window system. Block as you would XI 1.
161	UDP	snmp	Block.
162	UDP	snmp-trap	Block, unless you monitor routers outside of your net.
177	UDP	xdmcp	For XI I logins. Block, of course.
512	TCP	exec	Block. It could be useful with a variant rcp; as is, the only thing that has ever used it is the Internet worm. Besides, it doesn't do any logging.

513	TCP	login	Shudder Block.
514	TCP	shell	Double shudder It doesn't do any logging, either. Block.
515	TCP	printer	There have been reports of problems, and there's rarely a good reason for outsiders to use your printers. Block.
512	UDP	exec	Block; it's a buggy, dangerous service.
513	UDP	who	You shouldn't get anything legitimate on this port; block it.
514	UDP	syslog	Apart from security holes (and there are some), if this is open, your logs can be attacked. Block.
517	UDP	talk	Block; the actual protocol involves a conversation between random TCP ports.
518	UDP	ntalk	Ditto.
520	UDP	route	Block; don't allow outsiders to play games with your routing tables.
540	TCP	uucp	Historically a dangerous service, and mostly obsolete on the Internet. Block.
1025	TCP	listener	The usual port for the System V Release 3 listener. An amazingly bad choice; if you have such machines, either change the listener port (it's a local option), or be sure to block incoming calls only to this port; you're sure to have outgoing calls using it.
2000	TCP	openwin	Like XI 1. Block.
2049	UDP	nfs	Block, and don't think twice.
2766	TCP	listen	The System V listener. Like tcpmux, but with more services. Block.
6000-6xxx	TCP	x11	Block the entire range of XI I ports.
6667	TCP	IRC	Block. Internet Relay Chat may or may not be a security risk per se (although there are a few dangerous options in IRC clients), but some channels, at least, attract the sort of network people who send out ICMP Destination Unreachable messages.

2 Les niveaux de sécurité du DOD

Les critères d'évaluation de la sécurité des systèmes informatiques définissent des moyens de contrer et de limiter les atteintes au bon fonctionnement des systèmes: ceux-ci ont parfois des conséquences considérables.

Il existe différentes normes de référence qui définissent des critères de classement en niveau de la sécurité d'un système informatique. Plusieurs organisations se sont occupé de fixer les critères pour ces classements. D'abord, le Department of Defense qui, en 1985, a mis au point les premiers critères dans leur livre "Trusted Computer System Evaluation Criteria" aux Etats Unis, suivi par les Allemands avec leur référence "IT-Security Criteria", la communauté Européenne "Information Technology Security Evaluation Criteria", et le Canada "The Canadian Trusted Computer Product Evaluation Criteria".

Tous ces documents font référence à un document de base qui porte le nom d'"Orange Book" dans lequel le DoD a défini 6 conditions fondamentales dérivées de l'objectif de base, dont 4 s'occupent du contrôle d'accès à l'information, et les deux autres assurent la crédibilité de l'accomplissement de ces contrôles dans le système.

Ces conditions sont:

1 . La **politique de sécurité**: il doit y avoir une politique de sécurité explicite et bien définie forcée par le système.

2. Les **marques**: chaque objet doit être associé à un "label" qui indique le niveau de sécurité de l'objet.

3. L'**identification**: chaque objet doit être identifié d'une façon unique et convaincante. Cette identification est nécessaire afin que la demande d'accès puisse être vérifiée.

4. La **responsabilité**: le système doit maintenir des actions complètes qui touchent à la sécurité. De telles actions comprennent l'introduction de nouveaux utilisateurs dans le système, changement du niveau de sécurité d'un objet ou d'un sujet et des tentatives d'accès non autorisés.

5. L'**assurance**: le système informatique doit contenir des mécanismes qui renforcent la sécurité, et il doit être possible d'évaluer l'efficacité de ces mécanismes.

6. La **protection continue**: les mécanismes qui implémentent la sécurité doivent être protégés contre les changements non autorisés.

Toutes ces exigences doivent être respectées par les systèmes. Les classes sont basées sur des divisions qui peuvent être divisées en catégories. Il existe 4 divisions principales: A, B, C et D. Dans chaque division, il y a des niveaux différents. Une description plus détaillée de ces niveaux est présentée dans ce qui suit:

Les quatre divisions de sécurité définis par le DOD se composent de niveaux ou classe(s). Les niveaux supérieurs doivent respecter les contraintes des niveaux inférieurs. Les critères ont été développés dans le but d'atteindre trois objectifs:

1.fournir aux utilisateurs un critère d'évaluation du degré de confiance d'un système informatique pour le traitement des informations sensibles;

2.fournir des conseils aux producteurs dans le but de satisfaire les exigences de sécurité pour les applications sensibles;
3.fournir une base pour la spécification des exigences de sécurité dans les spécifications d'acquisition.

Ces niveaux sont définis dans l'"Orange Book" comme suit:

• **Le niveau D** est le niveau de protection minimale. Il n'y a pas de notion de protection entre utilisateur dans les systèmes de classe D (p.e. : les micro-ordinateurs).

• **Le niveau C** représente la sécurité avec accès discrétionnaire. Ce niveau se découpe en deux classes:

•**Classe CI**: la sécurité à ce niveau est assez réduite. Les systèmes classés CI ne sont pas suffisamment protégés contre les intrusions extérieures; l'environnement fourni sépare les utilisateurs de leurs données. Les utilisateurs doivent être autorisés de protéger leurs propres données pour ainsi limiter l'accès des autres utilisateurs ou groupes d'utilisateurs. Ceci suppose une authentification (procédure *login/passwd*) et un outil de protection fondé sur la notion de propriété (permission d'accès par utilisateur, groupe ou autres utilisateurs).

•**Classe C2**: dans laquelle l'authentification est plus stricte. Un système de cette classe est muni aussi d'un mécanisme d'audit. Les utilisateurs sont responsables de leurs actions. De plus, les ressources du système sont soumises à un contrôle d'accès et la documentation doit être complète. (p.e.: le système SunOS Release 4. 1.).Le niveau C2 exige simplement un mécanisme de génération d'information d'audit sans rien mentionner à propos de l'analyse de cette information.

• **Le niveau B**: La sécurité est mandatée. Cette division est découpée en trois classes:

•**Classe BI**: en plus des exigences du niveau C2, la classe BI dans laquelle les objets contrôlés sont individuellement étiquetés, le système doit assurer le respect des étiquettes. Celles-ci ne sont pas modifiables par les utilisateurs. L'accès aux objets se base sur l'étiquette, celle-ci en détermine le droit d'accès. (p.e.: SunOS MLS Release 1.0);

•**Classe B2**: dans laquelle:

1) la sécurité doit être représentée par un modèle;
2) il doit être possible de tester les outils de sécurité disponibles. Le système doit être associé à une notion de privilège minimal. Un des rares systèmes classés au niveau B2 est le système MULTICS de Honeywell Information System;

•**Classe B3**: est appelée "Domaines de sécurité". La structure interne du système doit être complètement documentée et prouvée informellement. Les listes de contrôle d'accès sont obligatoires. L'administration de la sécurité doit être une tâche séparée de l'administration du système lui-même. Dans cette classe, le système est très résistant aux tentatives d'intrusion.

•**Le niveau A**: La seule classe de la division A est la classe AI. Une preuve formelle est demandée. Il faut un modèle pour la protection du système et une preuve pour sa consistance et son adéquation, une spécification formelle de haut niveau du système de protection, une démonstration de la correspondance des spécifications de haut niveau au modèle, une implémentation informelle correspondant aux spécifications et une analyse formelle des canaux secrets.

3 LA SYNTAXE ET LA SEMANTIQUE DU LANGAGE RUSSEL

3.1 Le langage RUSSEL.

RUSSEL est un langage de règles permettant d'analyser tout fichier séquentiel structuré, et en particulier les fichiers NADF générés par l'adaptateur de format (voir troisième partie).

Une règle est considérée comme une procédure, au sens classique, avec des paramètres et des variables locales. Les paramètres permettent le passage d'informations entre les appels successifs des règles et donc entre les enregistrements.

Trois types de règles sont à distinguer:

1.**des règles d'initialisation:** elles sont activées au début du traitement, pour le premier enregistrement. Leur but est d'initialiser le processeur d'analyse en activant des règles pour le premier enregistrement;

2.des règles de sélection: ce sont des règles actives pendant l'évaluation d'un enregistrement donné. A chaque instant, cet ensemble de règles peut être modifié;

3.des règles de terminaison: elles sont évaluées après la fin du traitement du fichier de traces. Leur déclenchement peut activer, par exemple, l'impression de rapports sur les résultats de l'analyse, de la fermeture des fichiers, etc.

Une règle sert à détecter un événement dans un scénario, elle renferme un ensemble de conditions et une séquence d'actions. Les actions se présentent sous deux formes:

1.des actions spéciales basées sur un mécanisme de déclenchement de règles *(trigger off),* le déclenchement d'une nouvelle règle consiste à donner son nom, les valeurs des paramètres et un paramètre spécial indiquant si la règle sera activée pour le record courant **(trigger off for_current),** le record suivant **(trigger of for_next)** ou la fin de l'analyse **(trigger off at_completion).** Après avoir évalué un enregistrement, la règle est automatiquement désactivée, à moins d'être réactivée explicitement par elle-même ou à l'aide d'une autre règle;

2.des actions classiques sous forme d'appels de procédure prédéfinie permettant d'implémenter des fonctions de haut niveau (envoyer un message, déclencher une alarme, générer des fichiers temporaires, etc.).

Nous pouvons utiliser les règles pour détecter des tentatives d'intrusion et de violation de sécurité, ou pour faire des statistiques et des contrôles sur le taux d'utilisation des ports, des machines, etc. On peut supposer, pour une violation ou une infraction quelconque, un symptôme simplifié. Un ensemble de règles RUSSEL peut ainsi être proposé pour détecter ce symptôme. Ceci se fait en analysant le fichier NADF des traces. Les requêtes feront des filtrages multiples et variés pour détecter les informations contenues dans les messages envoyés sur le réseau.

L'objectif n'étant pas de développer toutes les sélections possibles, complète et sophistiquée mais plutôt de montrer la façon dont RUSSEL analyse ces informations (bytes captés) en filtrant sur les champs pertinents.

3.2 Types de données

Généralement deux types de donnée sont considérés: string de bytes; entier.

Toute donnée peut être représentée par un string de bytes. Les types de donnée doivent être le plus simple possible pour pouvoir assurer une compatibilité avec la grande variété de formats des fichiers de trace. ASAX utilise principalement le string de bytes; le type de donnée entier est parfois utilisé pour des opérations arithmétiques nécessaires pour spécifier des opérations de sélection complexes sur les *"data records"*

3.3 Syntaxe du langage RUSSEL.

3.3.1 Items lexicaux.

Définition des items lexicaux ou tokens sous forme BNF:

<token > ::= <identifier > | <constant > | <spécial symbol >
<identifier > ::= <letter >
| <identifier > <digit >
| <Identifier > <letter >
| <identifier > <underscore > <letter >
| <Identifier > <underscore > <digit >

<constant > ::= <integer constant >
| <C-literal >
| <X-literal >

<integer constant > ::= <digit >
| <integer constant > <digit >

<C-literal >::='<C_sequence >'
<C_sequence > :: = <empty >
| <C_sequence > <C-character >

<C-character > ::= any printable ASCII character except simple quote (')| "
<X-literal > ::= X '<X-sequence >'

<X-sequence > ::= <empty > | <X-sequence > <X-symboi >
<X-symbol > ::= <X - character > <X - character >
<X-character > ::= 0 | ... | 9 | A | B | C | D | E | F

<digit >::= 0 | | | 2 |3| 4 |5| 6 | 7 |8| 9 <letter > ::= a | ... | z | A | ... | Z

<special symbol >::= + | - | * | (|) | > | < | != | <= | >= | : | %= | ; | = | -> | := | , | and | at - completion | begin | div | do | end | false | fi | for-current | for-next | not | od | off | or | present | rule | skip | string | if | integer | mod | trigger | true | var

Le programme ASAX est constitué d'une séquence finie d'items lexicaux. Cette séquence doit respecter les règles syntaxiques suivantes:

3.3.2 Syntaxe abstraite.

La syntaxe abstraite du langage RUSSEL est la suivante:

<rule déclaration > ::= <rule heading ><variable déclaration part ><action part >
<rule heading > ::= **rule** <rule name > (<parameter list >)

<rule name > ::= <identifier >

<parameter list > ::= <empty >
| <parameter group > ; ... ;<parameter group >

<parameter group > <parameter name >,..., <parameter name > : <type>
<parameter name> : := <identifier >

<type > ::= **ascii-string | integer**

<variable déclaration part > ::= <empty >
| **var** <variable déclaration >; ... ; <variable déclaration >

<variable déclaration > ::= <variable name >, ...,<variable name > <type >

<variable name > ::= <identifier >
<action part > ::= <action >

<action > ::= **skip**
| <assignment >
| <conditional action >
| <repetitive action >
| <compound action >
| <rule triggering >
| <predefined procédure call >

<assignment > ::= left expression > := <right expression >

<left expression > ::= <parameter name > | <variable name >
<right expression > ::= <expression >

<expression > ::= <constant >
| <field name >
| <left expression >
| <expression > <binary arithmetic operator > <expression >
| <unary arithmetic operator > <expression >

<field name > ::= <identifier >

<conditional action > ::= **if** <guarded action >; ... ; <guarded action > **fi**

<guarded action > ::= <condition > -> <action >
<condition > ::= **true | false | present** <field name>
| <expression > <relational operator > <expression >
| <condition > <logical operator > <condition >
| **not** <condition >

<relational operator > ::= < | >
<repetitive action > ::= **do** <guarded action ><guarded action > **od**

<compound action > ::= **begin** <action > ;...; <action > **end**
<rule triggering > ::= **trigger off** <triggering mode > <rule call >
<triggering mode > ::= **for_next | for_current | at-Completion**
<predefined procédure call > ::= <predefined procédure name > «expression >, ... <expression »
<predefined procédure name > ::= <identifier >
<rule call > ::= <rule name > (<expression > ,...,<expression>)

<rule name > ::= <identifier >

La correction syntaxique complète d'une expression, d'une condition, ou d'une construction dépend de la table d'identifiants qui regroupe les noms de paramètres, les noms de variables et les noms des champs. Cette table s'appelle *identifier table.*

3.3.3 Syntaxe concrète.

La syntaxe concrète du langage RUSSEL est la suivante:

<condition > ::= <disjonction >
<disjonction > ::= <conjonction > | <disjonction > **or** <conjonction>
<conjonction > ::= <simple condition > | <conjonction > **and** <simple condition >

<simple condition > ::= **true** | **false** | **présent** <field name > | <relational expression > | (<condition >) |
not <condition >

<relational expression > ::= <arithmetic expression > <relational operator >
<arithmetic expression >

<relational operator > ::= < | > | # | = | <= | >=
<arithmetic expression > ::= <term >
| <arithmetic expression > <additive operator > <terrn >

<term > ::= <factor > | <term > <multiplicative operator ><factor >
<factor > ::= <simple expression >
<simple expression > ::= <Parameter name > | <variable name > | <integer - constant > | (<arithmetic
expression >) | <pre-defined procédure call >

<multiplicative operator > ::= * | **div** | **mod**
<additive operator > ::= + | -

La priorité des opérateurs de cette syntaxe concrète est:

Priority	*Operators*
1	*, **mod, div**
2	+ , -
3	<, >, <=, >=,<>
4	**not,present**
5	**and**
6	**or**

La priorité des opérateurs est implicite à la syntaxe concrète. Ces niveaux de priorité sont choisis de façon à ce que les expressions puissent être évaluées normalement.

3.4 Sémantique du langage RUSSEL.

Evaluation des expressions.
L'évaluation des expression est de quatre types:
1.Une constante littérale (C-littéral ou X-littéral) est évaluée au string du bytes qu'elle représente.

2.Une expression de la forme "**présent** *fieldname"* est dévaluée à la valeur VRAI si l'enregistrement courant contient le champ fieldname, sinon elle est évaluée à FAUX.

3.L'évaluation des expressions arithmétiques et relationnelles est la même que pour les autres langages de programmation connus.

4.De même pour les expressions conditionnelles simples et composées.

3.4.1 Exécution des actions.

Cinq types d'exécutions d'actions sont disponibles:
1. La construction **skip:** elle correspond à l'action vide. Son utilisation permet de clarifier le code d'une règle.

2. L'assignation: l'exécution de l'action *lexpr:=rexpr* par rapport à l'environnement courant revient à, d'abord, évaluer l'expression *rexpr* par rapport à l'environnement courant, puis affecter la valeur trouvé à la variable *lexpr.*

3. L'action conditionnelle: c'est l'exécution, par rapport à l'environnement courant, de l'action suivante:
if
cond 1 → action 1;
...
cond n → action n
fi

La première condition est évaluée; si elle est vrai, l'action « *action 1* » est exécutée par rapport à l'environnement courant et l'exécution est terminée; Sinon:
a) si n > 1 l'exécution reprend à:
if
cond 2 → action 2;
...
cond n → action n
fi

b) sinon l'exécution est terminé.

4. L'action répétitives: l'exécution de l'action répétitive, par rapport à l'environnement courant, est effectuée de la façon suivante:

do
cond 1 → action 1;
...
cond n → action n
od

Les conditions cond 1,.... *cond n* sont successivement évaluées par rapport à l'environnement courant jusqu'à ce que l'une d'entre elles soit évaluée à VRAI. Soit *cond i* cette condition. Dans ce cas, l'action *action i* est exécutée et ensuite l'action répétitive de départ est exécutée à nouveau; si aucune des conditions n'est évaluée à VRAI, l'exécution de l'action répétitive est terminée.

5.Les actions composées: l'exécution, par rapport à l'environnement courant, de l'action:
begin *action 1; ... ;action n* **end**
consiste à exécuter successivement les *actions i* par rapport à cet environnement.

3.4.2 Déclenchement de règles.

Soit E un environnement courant et *rule - name* une règle. L'exécution de l'action:
trigger off *tr-mode rule-name (expr 1,... , expr n);* (où n > 0, *et tr-Mode (triggering mode)* signifie mode de déclenchement) est effectué comme suit:

1.Les expressions expr |, ... *expr n* sont évaluées par rapport à l'environnement courant. Soit L = v1,... , vn = la liste de leurs valeurs respectives;

2.L'effet de l'exécution est d'ajouter cette règle à l'ensemble des règles actives:
- •pour le record courant si *tr -mode* **for_current;**
- •pour le record suivant si *tr-Mode* **for_next;**
- •à la fin de l'analyse si *tr-Mode* **at_Completion.**

3.4.3 Exécution d'un appel de procédures prédéfinies.

Soit proc-name une procédure prédéfinie, E l'environnement courant. La syntaxe de l'exécution de l'appel de procédure prédéfinie est la suivante: *proc-name* (expr 1,... expr n), avec n >= 0. Pendant l'exécution, on évalue d'abord les expressions expr 1,... expr n, par rapport à E, puis on passe leurs valeurs respectives à la procédure: *proc-name* v 1,... v n). Cette procédure est enfin exécutée par rapport à E (environnement courant).

3.4.4 Mécanisme de traitement et l'algorithme général de traitement.

3.4.4.1 Environnement initial

Supposons qu'on dispose, dans l'étape d'initialisation, des n règles actives suivantes: *RI1, ... RIn,* et, à la fin, des p règles actives suivantes: RCI, ... *RCp* . Supposons, également, que nous avons la règle suivante:

rule init-rule;

begin
Trigger off for-next RI1;
...
Trigger off for-next RIn;
Trigger off at-completion RC1;
...
Trigger off at_completion RCp; **end;**

L'environnement initial correspondant à ces règles d'initialisation et de fin de traitement est formé. Les composants suivants constituent cet environnement:
- le premier enregistrement du fichier de traces;
- l'environnement local (vide) de l'environnement initial;
- DStrig = { *init-rule };*
- DSnext = {};
- DScompl = {}.

Il est à noter que:
1. L'environnement initial devrait comprendre une description des procédures prédéfinies contenues dans la librairie ainsi que la description de quelques variables globales telle que la table du format adaptateur. Cependant, pour des raisons de simplicité, ces composants de l'environnement initial sont faits implicitement.
2. L'environnement initial devrait être définie comme suit:
- le premier enregistrement du fichier de traces;
- un environnement local vide;

- DStrig = { *RII, ..., RIn };*
- DSnext = {};
- DScompl = {*RCI, ... RCp}* .

3.4.4.2 Algorithme général de traitement.

Les *inputs* de cet algorithme sont:

•le fichier de traces sous forme NADF;
•un ensemble de définitions de règles composé de trois parties:
1.DSinit: ensemble de définitions de règles d'initialisation;
2.DStrig: ensemble de définitions de règles, correspondant à des règles devant êtredéclenchées dynamiquement;
3. DScompl: ensemble de définitions de règles de fin de traitement.

Les outputs sont : tous types de rapports générés pendant le traitement.

Au début de l'algorithme, DStrig est vide. La règle d'initialisation est formée des éléments de DSinit et DScompl.

Algorithme.

Initialisation:

Scurr:= init-rule;
Scompl:= {};
Ouvrir le fichier de traces;

En effet, au départ, la seule règle active est la règle de nom réservée **init-rule** alors que l'ensemble des règles actives à la fin de l'analyse est vide.

condition de fin de boucle:

fin du fichier de traces NADF

Itération:

lire l'enregistrement suivant;
traiter Scurr (en analysant l'enregistrement courant) pour produire NSnext et NScompl;
Scurr:= NSnext;
Scomp:= NScompl u Scopml;

En général, l'exécution des règles actives pour l'enregistrement courant peut activer d'autres règles pour l'enregistrement suivant et peut aussi activer d'autres règles pour la fin de l'analyse. Ces derniers viennent s'ajouter à l'ensemble des règles actives à la fin.

Clôture:

fermer le fichier de traces NADF;
exécuter les règles actives de fin de traitement.

4 Programme de l'adaptateur off-line.

```
#include <stdio.h>
#include <string.h>
FILE *ptrfile1;
int ptrfile2,cl2,wrtfile;
unsigned int i;
int c,ii;
char buffer[50];
char proto;
char first_code,second_code;
unsigned int src_addr_part[4],dest_addr_part[4];
unsigned int hour,min,sec;
unsigned int src_port,dest_port;
unsigned long code_nbr;
char *NADF_buffer_ptr,*char_ptr;
char NADF_buffer[512];
unsigned short *int_ptr;

/*-------------------------------------------------------------------------------------------------------- */
/*                                  PROGRAMME PRINCIPAL                                                   */
/*          Ce programme est constitu, essentiellement d'une boucle                                       */
/*           principale dans laquelle on parcoure le fichier source                                       */
/*          ligne par ligne en effectuant les op,rations propres ...                                      */
/*          chaque champ.                                                                                 */
/*-------------------------------------------------------------------------------------------------------- */

main (argc, argv)
char **argv;
{
 puts ("\n\t Ex,cution de l'adaptateur de format NADF");
 printf ("\t Fichier Source : %s\n", argv[1]);
 printf ("\t Fichier Cible  : %s\n", argv[2]);

 if (argc != 3)
          {
           puts("Vous devez entrer un nom de fichier source et un nom de fichier destination");
           exit ();
          }

 if ( (ptrfile1 = fopen (argv[1],"r")) == NULL)
          {
           printf ("\n\t %s ne peut pas ^tre ouvert\n", argv[1]);
           exit ();
          }

 ptrfile2=creat_NADF(argv[2]);
 printf("ptrfile2 =%d\n",ptrfile2);
 if ( ptrfile2 <0 )
          {
           printf ("\n\t %s ne peut pas ^tre ouvert\n", argv[2]);
           exit ();
          }
```

```
            /*  BOUCLE PRICIPALE     */

c=getc(ptrfile1);
while (  c  != EOF)
        {
         hour=parse_int_until(':');
         min=parse_int_until(':');
         parse_until(' ');

         src_addr_part[0]=parse_int_until('.');
         src_addr_part[1]=parse_int_until('.');
         src_addr_part[2]=parse_int_until('.');
         src_addr_part[3]=parse_int_until(' ');

         dest_addr_part[0]=parse_int_until('.');
         dest_addr_part[1]=parse_int_until('.');
         dest_addr_part[2]=parse_int_until('.');
         dest_addr_part[3]=parse_int_until(' ');

         proto=c;
         parse_until(' ');

         parse_code();

         src_port=parse_int_until(' ');
         dest_port=parse_int_until('\n');

         fill_buffer();

        }

fclose (ptrfile1);
cl2=close_NADF(ptrfile2);
printf("cl2 =%d\n",cl2);
return 0;
}
/*-------------------------------------------------------------------------------------------*/
/*                              Fonction PARSE_INT_UNTIL                                     */
/*        Cette fonction parcoure les caractŠres du fichier source                           */
/*        et les place dans le buffer sous forme d'entier non-sign,s.                        */
/*        Le parsing se fait jusqu'… un caractŠre donn,.                                     */
/* ------------------------------------------------------------------------------------------*/

int parse_int_until(ponct)
char ponct;
{
ii=0;
i=0;
buffer[0]=0;
while (c != ponct && c != EOF)
        {
        buffer[ii]=c;
        buffer[ii+1]=0;
        c=getc(ptrfile1);
        ii=ii+1;
        }
```

```
i=atoi(buffer);
c=getc(ptrfile1);
buffer[0]=0;
return i;
}

/* ------------------------------------------------------------------------------ */
/*                Fonction PARSE_UNTIL                                            */
/*   Cette fonction parcoure les caractŠres du fichier source                     */
/*   jusqu'... un caractŠre donn, sans les placer dans le buffer.                 */
/*------------------------------------------------------------------------------ */

int parse_until(ponct)
char ponct;
{
while (c != ponct && c != EOF )
        {
        c=getc(ptrfile1);
        }
c=getc(ptrfile1);
return 0;
}

/* ------------------------------------------------------------------------------*/
/*                Proc,dure PARSE_CODE                                           */
/*    Cette fonction parse les champs contenant les informations                 */
/*       relatives au CODE de TCP. Cette fonction permet de faire                */
/*       la diff,rence entre les paquets TCP et UDP.                             */
/*------------------------------------------------------------------------------*/

parse_code()
{
if (proto=='U')
        {
        first_code='X';
        second_code='X';
        code_nbr=0;
        }
if (proto=='T')
        {
        first_code=c;
        c=getc(ptrfile1);
        if (c == 32)
                {
                second_code='X';
                code_nbr=1;
                }
        else
                {
                second_code=c;
                code_nbr=2;
                }
        }
parse_until(' ');
return 0;
}
```

```
/*-------------------------------------------------------------------------------*/
/*                        Fonction FILL_BUFFER                                   */
/*-------------------------------------------------------------------------------*/
fill_buffer()

{
unsigned long real_length;
unsigned int a;
unsigned long *real_length_ptr;
unsigned int *longueur_ptr;

int inc;

NADF_buffer[0]=0;
NADF_buffer_ptr=NADF_buffer;
real_length_ptr=(unsigned long *) NADF_buffer;
real_length=82+6*(code_nbr);
*real_length_ptr=real_length;

NADF_buffer_ptr=NADF_buffer_ptr+4;

bufferize_char(100,proto);

for (inc=0 ; inc<4 ; inc++)
        {
        bufferize_int(101+inc,src_addr_part[inc]);
        }

for (inc=0 ; inc<4 ; inc++)
        {
        bufferize_int(105+inc,dest_addr_part[inc]);
        }

bufferize_int(109,src_port);
bufferize_int(110,dest_port);
bufferize_int(111,hour);
bufferize_int(112,min);


if (real_length>82)
        {
        bufferize_char(113,first_code);
        if (real_length>88)
                {
                bufferize_char(114,second_code);
                }
        }

NADF_buffer_ptr=NADF_buffer;
if (( wrtfile = write_NADF(ptrfile2,NADF_buffer_ptr,1)) != 0)
        printf("Impossible de cr,er le fichier NADF %d\n", wrtfile);
return 0;
}
```

```
/*-------------------------------------------------------------------------------------------------------------*/
/*                                  Fonction BUFFERIZE_INT                                                     */
/*        Cette fonction remplit le buffer NADF avec un unsigned int,                                          */
/*        son identifiant et sa longueur en byte.                                                              */
/*-------------------------------------------------------------------------------------------------------------*/
bufferize_int(id,val)
unsigned int id,val;
{
int_ptr=(unsigned short *)NADF_buffer_ptr;
*int_ptr=id;
NADF_buffer_ptr=NADF_buffer_ptr+2;
int_ptr=(unsigned short *)NADF_buffer_ptr;
*int_ptr=2;
NADF_buffer_ptr=NADF_buffer_ptr+2;
int_ptr=(unsigned short *)NADF_buffer_ptr;
*int_ptr=val;
NADF_buffer_ptr=NADF_buffer_ptr+2;
return 0;
}
/*-------------------------------------------------------------------------------------------------------------*/
/*                                  Fonction BUFFERIZE_CHAR                                                    */
/*        Cette fonction remplit le buffer NADF avec un caractŠre,                                             */
/*        son identifiant et sa longueur en byte.                                                              */
/*-------------------------------------------------------------------------------------------------------------*/
bufferize_char(id,val)
unsigned short id;
char val;
{
int_ptr=(unsigned short *) NADF_buffer_ptr;
*int_ptr=id;
NADF_buffer_ptr=NADF_buffer_ptr+2;
int_ptr=(unsigned short *) NADF_buffer_ptr;
*int_ptr=1;
NADF_buffer_ptr=NADF_buffer_ptr+2;
char_ptr=NADF_buffer_ptr;
*char_ptr=val;
NADF_buffer_ptr=NADF_buffer_ptr+2;
return 0;
}
```

5 programme awk.

```
$1 ~ /[0-9][0-9]\:[0-9][0-9]\:/ \
  {d_time = $1;
   split($2,tab,".")
   ip_src = tab[1]."tab[2]."tab[3]."tab[4];
   split(tab[5],tob,":");
   src_port = tob[1];
   split($4,tab,".");
   ip_dst = tab[1]."tab[2]."tab[3]."tab[4];
   split(tab[5],tob,":");
   dst_port = tob[1];
   if ( $5 == "F" || $5 == "S" || $5 == "FP"){
     flags = $5;
     printf("%s %s %s TCP %s %s
%s\n",d_time,ip_src,ip_dst,flags,src_port,dst_port);
    }
   else {
     printf("%s %s %s UDP X %s %s\n",d_time,ip_src,ip_dst,src_port,dst_port);
    }
  }
```

6 Exemple de règle RUSSEL n°1

Cette règle permet l'affichage du contenu du fichier NADF dans sa totalité.

```
init_action;
trigger off for_next display;

rule display;
begin
        println(HOUR,':',MIN,'--> from
:',SRC_ADDR_PART_1,'.',SRC_ADDR_PART_2,'.',SRC_ADDR_PART_3,'.',SRC_ADDR_PART_
4,'
to:',DEST_ADDR_PART_1,'.',DEST_ADDR_PART_2,'.',DEST_ADDR_PART_3,'.',DEST_ADDR_
PART_4);
        trigger off for_next display;
end.
```

7 Exemple de règle RUSSEL n°2

La règle ci-dessous permet de surveiller l'utilisation de port TCP sensibles comme :

43	TCP	whois
53	UDP, TCP	DNS
70	TCP	gopher
79	TCP	finger
161	UDP	SNMP
2000	TCP	Openwin
2049	UDP	NFS
6000-6xxx	TCP	X11
6667	TCP	IRC

```
init_action;
        trigger off for_next port_warning;

rule port_warning;
begin
        if
                        (
                        PROTOCOL=T
                        and
                        (SCR_PORT=43 or SCR_PORT=53 or SCR_PORT=70
                                or SCR_PORT=79 or SCR_PORT=161)
                        )
                        or
                        (
                        PROTOCOL=T
                        and
                                (
                                (SCR_PORT=2000 or DEST_PORT=2000)
                                or
                                (SCR_PORT=6667 or DEST_PORT=6667)
                                or
                                (
                                (SCR_PORT>=6000 and SRC_PORT<=7000)
                                                or
                                (DEST_PORT>=6000 and DEST_PORT<=7000)
                                )
                                )
                        )

                --> trigger_off for_next warning_message(SRC_PORT, DEST_PORT ,HOUR,
MIN, SRC_ADDR_PART_1, SRC_ADDR_PART_2, SRC_ADDR_PART_3,
SRC_ADDR_PART_4, DEST_ADDR_PART_1, DEST_ADDR_PART_2, DEST_ADDR_PART_3,
DEST_ADDR_PART_4 ) ;
        fi;
        trigger off for_next port_warning ;
end;
```

```
rule warning_message(s_port, d_port, hr, mn, src_part1, src_part2, src_part3, src_part4, dest_part1,
dest_part2, dest_part3, dest_part4 :integer) ;
begin
        println('----------------------------------------');
        println(s_port, d_port, hr, mn, src_part1, src_part2, src_part3, src_part4, dest_part1,
dest_part2, dest_part3, dest_part4 );
        println('----------------------------------------');
end.
```

NAME

vopen, vread, vclose - user provided virtual input output routines on streams.

SYNOPSIS

```
int vopen (argc, argv)
int argc;
char **argv;

int vread (ptr)
char **ptr;

int vclose
```

DESCRIPTION

vopen:

Precondition

argv s a pointer to an array of character strings that contains arguments one per string. These are the remaining arguments after the module name argument in ASAX command line. argc is the number of these arguments (>=0).

This means that ASAX's main routine passes down the rest of its command line arguments to vopen.

Postcondition

A stream of records is opened for reading i.e., the next call to vread(). is able to read the first record in the stream if there is one.

vread:

Precondition

A stream was previously opened as in the postcondition of vopen().

Postcondition

*ptr is the address of the next record in the stream. This record has the NADF format (see the ASAX user guide).

vclose:

Precondition

A stream was previously opened as in the postcondition of vopen().

Postcondition

The stream is closed.

RETURN VALUES

vopeno, vreado and vcloseo all return:

0 on success.

-1 on failure. in this case, ASAX exits completely.

Bibliographie

Pour la **première partie** :

T. Parker, « TCP/IP », s&sM, 1996.

D. Comer, « Internetwiorking With TCP/IP », Prentice-Hall, 1988.

Pour la **deuxième partie** :

D.A.R.P.A. « Transmission Control Protocol » ; RFC 793 ; septembre 1981.

S. Bradner, A. Mankin ; « Recommandations for IP next generation protocol » ; RFC 1752 ; janvier 1995

W. R. Cheswick, S. M. Bellovin , « *Firewall*s ans Internet Security », ADDISSON-WESLEY, 1994.

S.M. Bellovin, « Security Problems in the TCP/IP Protocole Suite », Computer Communication Review, Vol. 19, n°2, pp. 32-48, avril 1989.

S.M. Bellovin, « Packets Found on an Internet », Computer Communication Review, Vol. 23, n°3, pp. 26-31, août 1993.

L. Joncheray, « A Simple Effective Attack Against TCP », avril 1995.

Deamon9, route, infinity, « IP-Spoofing Demystified », Phrack Magazine, Vol. 7, Issue 48, juin 1996.

B. Guha, B. Mukherjee, « Network Security Via Reverse Engineering of TCP Code : Vulnerability and Proposed Solutions », IEEE proceedings, Conference Infocom 1996.

R. Braden, « Time-WAIT Assassination Hazards in TCP », RFC 1337, mai 1992.

J. Reynolds, « The Helminthiasis of the Internet », RFC 1135, decembre 1989.

A. Muffett, « Peoper Care and Feeding *Firewall*s », Sun Microsystems, novembre 1994.

B. Corbridge, R. Henig, C. Slater, « Packet Filtering in an IP Router », LISA V, sep. 30- oct. 3, 1991.

ASAX (Advanced Sequential File Analysis product) ; Software development project file ; Siemens Nixdorf Software S .A. Namur ; rev. 2 ; 1994.

Pour la **troisième partie** :

S. Bourne, « Le système Unix », InterEditions, 1985.

E. Nemeth, G. Snyder, S. Seebass, « Unix Administration Handbook », Prentice-Hell Software Series.

Autre publication concernant ASAX :

A. Mounji ; « Rule_based distibuted Intrusion Detection » ; Institut d'informatique ;Facultés universitaires Notre Dame de la Paix Namur ; (E-mail :amo@info.fundp.ac.be); juillet 1997